Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно–інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інформаційного сайту для вивчення технологій веб-дизайну та веб-програмування

Виконав: студент IV курсу, групи СН–41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Карпінський О.Р.
(підпис) (прізвище та ініціали)

Керівник _____ Дуда О.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Тимощук Д.І.
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно–інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_27_» _січня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Карпінському Олександру Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інформаційного сайту для вивчення технологій веб-дизайну та веб-програмування

Керівник роботи Дуда Олексій Михайлович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*07* « *травня* 2025 року № *4/7–444*

2. Термін подання студентом завершеної роботи 25 червня 2024р.

3. Вихідні дані до роботи Інформаційні матеріали та джерела для вивчення технологій веб-дизайну та веб-програмування

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області веб-дизайну та веб-програмування. 1.1 Веб дизайн: поняття, принципи та стандарти. 1.2 Веб-програмування: технології фронтенд та бекенд. 1.3 Висновок до першого розділу. 2 Проектна частина. Архітектура та функціональність інформаційних сайтів. 2.1 Клієнт-серверна архітектура веб-застосунків. 2.2 Фунціональні особливості та вимоги до інформаційних навчальних сайтів. 2.3 Висновок до другого розділу 3 Практична частина. Реалізація інформаційного сайту для вивчення веб-технологій. 3.1 Загальна структура сайту та засоби реалізації. 3.2 Приклад реалізації фронтенду (HTML та CSS). 3.3 Приклад реалізації бекенду (PHP). 3.4 Висновок до третього розділу 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу
1 Титульна сторінка. 2 Тема, Мета, Об'єкт, Предмет дослідження. 3 Завдання дослідження. 4 Актуальність дослідження. 5 Рівні моделі користувацького досвіду. 6 Основні технології веб-розробки. 7 Структура HTML-документа. 8 Клієнт–серверна архітектура веб-застосунку. 9 Інформаційна архітектура навчального сайту. 10 Макет адаптивного інтерфейсу сайту. 11 Загальна структура папок та файлів сайту. 12 Фрагмент шаблонів. 13 Вигляд Головної сторінки сайту. 14 Вигляд сторінки Уроку. 15 Висновки. 16 Завершальний.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С. к.т.н., доц. кафедри МТ	02.06.2025	05.06.2025

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	*Виконано*
2.	Підбір джерел про розробки інформаційного сайту	08.05.2025-15.05.2025	*Виконано*
	для вивчення технологій веб-дизайну і веб-програм.		
4.	Виконання дослідження згідно мети кваліфікаційної	16.05.2025-22.05.2025	*Виконано*
	роботи		
5.	Оформлення розділу «Аналіз предметної області	23.05.2025-01.06.2025	*Виконано*
	веб-дизайну та веб-програмування»		
6.	Оформлення розділу «Проектна частина. Архітектура	23.05.2025-01.06.2025	*Виконано*
	та функціональність інформаційних сайтів»		
6.	Оформлення розділу «Практична частина. Реалізація	23.05.2025-01.06.2025	*Виконано*
	інформаційного сайту для вивчення веб-технологій»		
7.	Виконання завдання до підрозділу «Безпека	02.06.2025-05.06.2025	*Виконано*
	життєдіяльності»		
8.	Виконання завдання до підрозділу «Основи охорони	02.06.2025-05.06.2025	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	06.06.2025-10.06.2025	*Виконано*
10.	Нормоконтроль	11.06.2025-13.06.2025	*Виконано*
11.	Перевірка на плагіат	16.06.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	18.06.2025	*Виконано*
13.	Захист кваліфікаційної роботи	25.06.2025	

Студент _____ Карпінський О.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дуда О.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інформаційного сайту для вивчення технологій веб-дизайну та веб-програмування // Кваліфікаційна робота освітнього рівня «Бакалавр» // Карпінський Олександр Русланович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно–інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН–41 // Тернопіль, 2025 // С.64 , рис. – 9, табл. – 0, кресл. – 16, додат. – 8, бібліогр. – 42.

Ключові слова: веб-дизайн, html, css, php, інформаційний сайт, навчання, веб-програмування, інтерфейс.

Кваліфікаційна робота присвячена розробці інформаційного сайту для вивчення технологій веб-дизайну та веб-програмування.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»: Розглянуто ключові мови та технології, зокрема HTML, CSS і PHP; Висвітлено сучасні підходи до структурування контенту та побудови інтерфейсів;

В другому розділі кваліфікаційної роботи: Досліджено архітектурні рішення та принципи побудови інформаційних сайтів; Обґрунтовано вибір структури та функціонального наповнення ресурсу; Сформовано логічну модель сайту для навчання веб-технологіям.

В третьому розділі кваліфікаційної роботи: Розроблено прототип інформаційного сайту з основними сторінками та функціоналом; Спроєктовано зручну навігацію, адаптивне відображення та структуру уроків; Протестовано базову функціональність сайту, включно з формою зворотного зв'язку.

В четвертому розділі кваліфікаційної роботи: висвітлено питання з Безпеки життєдіяльності та Основ охорони праці.

Об'єкт дослідження: процес створення інформаційного веб-сайту як засобу навчання веб-технологіям.

Предмет дослідження: методи та інструменти розробки освітніх веб-ресурсів із використанням HTML, CSS та PHP.

ANNOTATION

Development of an Information Website for Studying Web Design and Web Programming Technologies // Qualification work of the educational level «Bachelor» // Karpinskyi Oleksandr Ruslanovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN–41 // Ternopil, 2025 // P.64, fig. – 9, tabl. – 0, chair. – 16, annexes. – 8, references – 42.

Keywords: web design, html, css, php, information website, training, web programming, interface.

The qualification work is devoted to the development of an information website for studying web design and web programming technologies.

In the first section of the qualification work of the educational level 'Bachelor': Key languages and technologies, in particular HTML, CSS and PHP, are considered; Modern approaches to content structuring and interface design are highlighted;

The second section of the qualification work: Architectural solutions and principles of building information websites are investigated; The choice of structure and functional content of the resource is justified; A logical model of the website for learning web technologies is formed.

In the third section of the qualification work: A prototype of an information website with main pages and functionality is developed; Convenient navigation, adaptive display and lesson structure are designed; The basic functionality of the website is tested, including the feedback form.

In the fourth section of the qualification work: issues of life safety and occupational safety basics are highlighted.

Object of research: the process of creating an informational website as a means of teaching web technologies.

Subject of research: methods and tools for developing educational web resources using HTML, CSS and PHP.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML (англ. HyperText Markup Language) – мова розмітки гіпертексту.

CSS (англ. Cascading Style Sheets) – каскадні таблиці стилів.

PHP (англ. PHP: Hypertext Preprocessor) – серверна скриптова мова програмування.

JS (англ. JavaScript) – мова програмування JavaScript.

UI (англ. User Interface) – користувацький інтерфейс.

UX (англ. User Experience) – досвід користувача.

HTTP (англ. HyperText Transfer Protocol) – протокол передавання гіпертексту.

HTTPS (англ. HyperText Transfer Protocol Secure) – захищений протокол HTTP.

URL (англ. Uniform Resource Locator) – уніфікований локатор ресурсу.

URI (англ. Uniform Resource Identifier) – уніфікований ідентифікатор ресурсу.

DOM (англ. Document Object Model) – об’єктна модель документа.

API (англ. Application Programming Interface) – інтерфейс прикладного програмування.

CMS (англ. Content Management System) – система керування контентом.

SQL (англ. Structured Query Language) – структурована мова запитів.

DB (англ. Database) – база даних.

CRUD (англ. Create, Read, Update, Delete) – базові операції з даними.

SEO (англ. Search Engine Optimization) – пошукова оптимізація.

MVC (англ. Model–View–Controller) – архітектурний шаблон «модель–представлення–контролер».

SPA (англ. Single-Page Application) – односторінковий веб-застосунок.

CDN (англ. Content Delivery Network) – мережа доставлення контенту.

SSL/TLS (англ. Secure Sockets Layer / Transport Layer Security) – протоколи захищеного з’єднання.

W3C (англ. World Wide Web Consortium) – Консорціум Всесвітньої павутини.

WCAG (англ. Web Content Accessibility Guidelines) – рекомендації з доступності веб-контенту.

LAMP (англ. Linux, Apache, MySQL, PHP/Python/Perl) – класичний стек веб-розробки.

REST (англ. Representational State Transfer) – стиль архітектури веб-служб.

JSON (англ. JavaScript Object Notation) – формат обміну даними «нотація об’єктів JavaScript».

XML (англ. eXtensible Markup Language) – розширювана мова розмітки.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний світ налічує величезну кількість веб-сайтів – станом на 2025 рік у світі існує понад 1,2 мільярда веб-сайтів [1]. Веб-технології продовжують стрімко розвиватися, а попит на фахівців з веб-дизайну та веб-програмування зростає. За прогнозами, зайнятість веб-розробників і цифрових дизайнерів у світі зросте приблизно на 8% протягом 2023–2033 років, що швидше за середній показник для всіх професій [2]. Це підкреслює актуальність навчання веб-технологіям і постійного вдосконалення навичок у цій галузі.

Одночасно стрімко зростає і роль онлайн-освіти. За останні два десятиліття масштаб навчання через інтернет зріс більш ніж на 900%, зробивши онлайн-освіту найдинамічнішим сегментом освітньої індустрії [3]. Все більше людей віддають перевагу самостійному вивченню IT-навичок через веб-ресурси. Згідно з опитуванням Stack Overflow 2023 року, 80% респондентів зазначили, що навчалися програмуванню за допомогою онлайн-ресурсів (відео, блогів, форумів тощо), порівняно з 70% роком раніше [4]. Це свідчить про зростання популярності відкритих онлайн-матеріалів для вивчення технологій. Наприклад, безкоштовна освітня платформа freeCodeCamp щоденно обслуговує понад мільйон користувачів, що опановують веб-програмування [5]. Таким чином, створення якісних інформаційних сайтів для навчання веб-дизайну і програмування є своєчасним та суспільно важливим завданням.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробити інформаційний веб-сайт, призначений для підтримки навчання технологіям веб-дизайну та веб-програмування. Для досягнення цієї мети в роботі вирішуються такі завдання:

– проаналізувати теоретичні основи веб-дизайну та веб-програмування, визначити ключові поняття, технології та принципи;

– дослідити архітектуру та функціональні можливості інформаційних сайтів, особливості їх структури та вимоги до UX/UI;

– спроєктувати та реалізувати практичну частину інформаційного сайту для навчання веб-технологіям, продемонструвати приклади коду HTML, CSS і PHP, перевірити працездатність основних функцій.

Практичне значення одержаних результатів. Розроблений інформаційний веб-сайт «WebTutor» має низку прикладних переваг, що дозволяють безпосередньо використовувати його у навчальній, професійній та суспільній діяльності. По-перше, ресурс забезпечує доступну платформу для самоосвіти: структуровані курси з HTML, CSS і PHP, доповнені прикладами коду, формують завершений освітній маршрут для початківців і студентів IT-спеціальностей. Завдяки відкритій ліцензії матеріали можуть інтегруватися у навчальні плани закладів вищої та фахової передвищої освіти, скорочуючи витрати викладачів на підготовку власних методичних розробок.

По-друге, запропонована модульна архітектура (шаблони header/footer, відокремлені стилі, компонентне меню) слугує універсальним каркасом для створення тематичних сайтів будь-якої освітньої спрямованості. Використання PHP-шаблонів і секційного контенту спрощує підтримку та дає змогу швидко розширювати функціонал (підключати базу даних, тестові модулі, особисті кабінети) без суттєвих змін коду. Це робить проект придатним як «reference implementation» у дисциплінах «Веб-програмування», «Системи керування контентом», «Інженерія програмного забезпечення».

По-третє, реалізація адаптивного дизайну, базових заходів безпеки (санітизація введених даних, валідація email) та форми зворотного зв’язку демонструє best-practice підходи до розроблення малих і середніх веб-застосунків. Такий живий приклад дозволяє студентам у процесі навчання оцінити вплив UX-рішень на залученість користувачів і опанувати типові серверні сценарії (надсилання email, динамічна генерація меню).

По-четверте, сайт може виступати інструментом популяризації цифрової грамотності: вільний доступ до навчальних матеріалів сприяє підвищенню фахових компетентностей населення, підтримує концепцію безперервного навчання (lifelong learning) та може бути використаний у програмах

перекваліфікації, зокрема для ветеранів або внутрішньо переміщених осіб, яким потрібні нові IT-навички.

Розробка створює підґрунтя для наукових і комерційних інновацій. Її результати можуть бути розширені у напрямі інтеграції систем автоматичного тестування, аналітики навчального прогресу або впровадження AI-помічників для адаптивного контенту. Таким чином, одержані результати мають реальний вплив на підготовку кадрів, розвиток освітньої інфраструктури та підвищення конкурентоспроможності фахівців у галузі веб-технологій.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ВЕБ-ДИЗАЙНУ ТА ВЕБ-ПРОГРАМУВАННЯ

1.1 Веб-дизайн: поняття, принципи та стандарти

Веб-дизайн – це галузь, що охоплює проектування візуального вигляду та взаємодії користувача з веб-сайтами. Веб-дизайнери зосереджуються на тому, як виглядає сайт (оформлення сторінок, кольори, типографіка, компонування), а також на зручності його використання. Результатом якісного веб-дизайну є інтуїтивно зрозумілий інтерфейс, який забезпечує позитивний досвід користувача (UX) і досягає поставлених цілей сайту. Кінцевий продукт веб-дизайну – це користувацький інтерфейс (UI) веб-сторінки, тобто розташування всіх елементів (навігація, тексти, зображення, кнопки тощо) у зручний для відвідувача спосіб.

Веб-дизайн спирається на низку принципів, вироблених як практикою, так і дослідженнями у галузі людсько-комп’ютерної взаємодії. Зокрема, ще у 1990-х роках експерт з юзабіліті Якоб Нільсен сформулював 10 евристичних правил зручності інтерфейсу – серед них принципи очевидності, відповідності очікуванням користувача, консистентності, запобігання помилкам тощо [6]. У своїй впливовій книзі “Don’t Make Me Think” Стів Круг проголосив перший закон юзабіліті: інтерфейс має бути настільки зрозумілим, щоб у користувача не виникало додаткових запитань – тобто “не змушуй мене думати” під час навігації сайтом. Іншими словами, добре спроектований сайт повинен бути самодокументованим: користувач одразу бачить, як виконати потрібні дії, без довгих роздумів чи інструкцій.

Дотримання принципів юзабіліті прямо впливає на успіх веб-ресурсу. Дослідження показують, що перше враження про сайт формується надзвичайно швидко – приблизно за 50 мілісекунд – і у 94% випадків базується саме на візуальному дизайні сторінки [8]. Якщо дизайн виглядає застарілим або

незручним, відвідувач схильний негайно втратити довіру до ресурсу. Крім того, поганий користувацький досвід (неінтуїтивна навігація, повільна робота, помилки) призводить до відтоку аудиторії: 88% користувачів не повертаються на сайт після невдалої взаємодії [8]. Ці статистичні дані підкреслюють важливість того, щоб веб-дизайн приділяв увагу зручності та привабливості інтерфейсу.

Серед ключових складових ефективного веб-дизайну можна виокремити такі принципи:

– Простота і зрозумілість: Інтерфейс має подавати інформацію простою мовою і логічною структурою. Елементи керування (посилання, кнопки) повинні однозначно підказувати свою функцію. Як зазначає Круг, сторінки слід проектувати так, щоби користувачі могли зрозуміти їх без додаткових роз'яснень – "сам собою зрозумілий" дизайн є ідеалом зручності.

– Візуальна ієрархія: Розташування та оформлення елементів повинно відображати їхню важливість. Наприклад, заголовки більші за розміром і виділені жирним шрифтом, основні кнопки мають яскраве забарвлення тощо. Правильна візуальна ієрархія спрямовує увагу користувача на ключовий контент.

– Консистентність (єдність стилю): Усі сторінки сайту бажано оформлювати в єдиному стилі, використовуючи спільні шаблони для навігації, кольорової схеми, типографіки. Консистентність полегшує користувачу навчання взаємодії з сайтом – він бачить знайомі елементи й не плутається.

– Адаптивність: Сьогодні значна частина веб-трафіку надходить з мобільних пристроїв – понад 62% глобального інтернет-трафіку генерують смартфони [27]. Тому дизайн має бути адаптивним (responsive), тобто підлаштовуватися під різні розміри екранів. Адаптивний веб-дизайн забезпечує зручність перегляду як на великих моніторах, так і на смартфонах, використовуючи гнучкі сітки верстки, відносні розміри елементів та медіа-запити CSS.

– Доступність: Веб-сайти повинні бути доступними для якомога ширшої аудиторії, включно з людьми з інвалідністю. Це означає, наприклад, що текст повинен бути достатньо контрастним, зображення – мати альтернативні описи, а навігація – бути зрозумілою і без використання виключно візуальних підказок. Ігнорування доступності призводить до втрати користувачів: близько 71% відвідувачів полишають сайт, якщо він виявляється складним для навігації людям з особливими потребами [8]. Дотримання стандартів доступності (наприклад, WCAG) не лише робить ресурс інклюзивним, але й часто покращує загальну зручність для всіх користувачів.

– Швидкодія та оптимізація: Користувачі цінують швидке завантаження сторінок та плавну роботу інтерфейсу. Повільний сайт може відлякати відвідувачів; згідно з дослідженнями, 8 з 10 користувачів покинуть сторінку, яка некоректно відображається або надто довго завантажується на їхньому пристрої [9]. Тому веб-дизайн повинен враховувати оптимізацію графіки, мінімізацію зайвих скриптів, використання кешування тощо для покращення продуктивності.

Важливим аспектом дизайну контенту є структурування та читабельність тексту. Користувачі, переглядаючи веб-сторінки, зазвичай сканують текст, видивляючись потрібну інформацію, а не читають слово в слово. Тому необхідно розбивати матеріал на невеликі абзаци, виділяти заголовки, списки, ключові терміни. Практика показує, що зручність сприйняття суттєво зростає, якщо текст зроблено сканованим: юзабіліті-метрики покращуються на 47% у разі, коли сторінка містить добре структурований, «сканований» текст [9]. Отже, веб-дизайн навчального сайту повинен приділяти особливу увагу структурі подачі матеріалу – щоб користувач легко орієнтувався, знаходив потрібні розділи й міг швидко виокремити головні думки.

У підсумку, теоретичні основи веб-дизайну зводяться до поєднання естетики та зручності. Хороший дизайн є користувацько-орієнтованим (user-centered) – тобто враховує потреби та можливості аудиторії. Як зазначає Джессі Джеймс Гарретт у моделі п’яти рівнів користувацького досвіду, успішний веб-

продукт починається зі стратегії та вимог (цілі й потреби користувачів), далі вибудовується структура (інформаційна архітектура та навігація), переходить у форму скелету (макети сторінок, розміщення елементів) і завершується поверхневим шаром – візуальним дизайном інтерфейсу [12]. Всі ці рівні мають бути узгоджені між собою. У цій роботі при створенні навчального сайту враховуються зазначені теоретичні принципи – від планування структури контенту до реалізації привабливого та інтуїтивного інтерфейсу.

Також на Рисунку 1.1 наведено п’ять рівнів моделі користувацького досвіду за Дж. Дж. Гарреттом. Схема відображає стратегію → вимоги → структуру → скелет → поверхню, показуючи зв’язок між логікою побудови продукту та його візуальним оформленням.

Рисунок 1.1 – П’ять рівнів моделі користувацького досвіду за Дж. Дж. Гарреттом

1.2 Веб-програмування: технології фронтенд та бекенд

Веб-програмування забезпечує функціональність веб-сайтів – тобто динамічну взаємодію, логіку роботи та обробку даних. У сучасній веб-розробці традиційно розрізняють дві сторони: front-end (клієнтська сторона) та back-end (серверна сторона) розробки. Front-end охоплює все, що відбувається в браузері

користувача: відображення сторінок, реагування на дії користувача, динамічні ефекти інтерфейсу. Back-end – це те, що відбувається “за лаштунками” на сервері: зберігання та обробка даних, бізнес-логіка, генерування сторінок на запит клієнта.

Для розробки front-end використовується комбінація трьох основних технологій: HTML, CSS та JavaScript. HTML (HyperText Markup Language) є стандартною мовою розмітки гіпертексту, що визначає структуру веб-контенту. По суті, HTML – це каркас сторінки, набір елементів (тегів), які означають, де знаходяться заголовки, абзаци, зображення, посилання тощо. HTML є базовим будівельним блоком Вебу і задає смислову структуру вмісту сторінки [14-15]. Історично HTML був створений британським ученим Тімом Бернерс-Лі наприкінці 1980-х – перша версія мови та протоколів Всесвітнього павутиння з’явилася вже у 1991 році, коли Бернерс-Лі опублікував опис 18 початкових тегів HTML. Відтоді HTML еволюціонував, включаючи нові можливості. Сучасний стандарт HTML5 визначає багато семантичних елементів (напр. <header>, <nav>, <article>, <section> та ін.), що дозволяють розробникам логічніше структурувати сторінки і покращувати розуміння контенту машинами (пошуковими системами) і людьми.

CSS (Cascading Style Sheets) – каскадні таблиці стилів – відповідає за оформлення та вигляд HTML-вмісту. CSS забезпечує відокремлення представлення від структури: якщо HTML описує що відображається, то CSS описує як це виглядає у браузері. Це декларативна мова стилів, якою задаються кольори, шрифти, розміри, розташування елементів на сторінці тощо [14-15]. Впровадження CSS дало змогу значно покращити дизайн сайтів і спростити підтримку – стилі можна визначити централізовано і застосувати до багатьох сторінок. Технологію CSS було запропоновано Гоконом Віум Лі у 1994 році (у співпраці з Бертом Босом і за підтримки консорціуму W3C), а вже у 1996 році вийшла перша рекомендація CSS1 [16]. Нині використовуються розширені версії CSS (рівні 2.1, 3), що включають десятки модулів – від модулю компоновки Flexbox і Grid до анімацій, адаптивного дизайну (медіа-запити) і

т.ін. CSS є однією з трьох основних технологій Вебу нарівні з HTML і JavaScript.

JavaScript (JS) – це мова програмування, що виконується на стороні клієнта (в браузері) і забезпечує інтерактивність веб-сторінок. Саме завдяки JavaScript реалізуються такі речі, як валідація форм на клієнті, динамічна зміна контенту без перезавантаження сторінки, різноманітні інтерактивні віджети, модальні вікна, а також односторінкові додатки (SPA) у сучасних фреймворках. JavaScript було вперше впроваджено компанією Netscape у 1995 році, і сьогодні ця мова підтримується усіма браузерами. Разом із HTML та CSS, JavaScript утворює «велику трійцю» фронтенд-розробки. У контексті нашого навчального сайту JavaScript міг би використовуватися для покращення взаємодії з користувачем (наприклад, приховування/розгортання відповідей на тестові питання, підсвічування синтаксису коду, перевірки вводу в формах). Проте основний акцент проекту зроблено на серверній логіці, тому використання JS мінімізоване (припустимо, тільки базові сценарії за потреби).

На бекенд (серверній) стороні веб-програмування функціонує програмне забезпечення, яке обробляє запити від клієнта, звертається до баз даних, бізнес-логіки і формує динамічний контент для відправки користувачу. Бекенд може бути реалізований різними мовами і технологіями. Одним з найпоширеніших рішень для серверного веб-програмування є мова PHP (PHP: Hypertext Preprocessor). PHP – це популярна скриптова мова загального призначення, спеціально пристосована для розробки веб-застосунків [17]. Вона була створена Рамусом Лердорфом у 1994 році і відтоді постійно розвивається [18]. Станом на 2025 рік PHP залишається домінуючою серверною технологією: її застосовують приблизно 79% усіх веб-сайтів, на яких відома мова з боку сервера [20]. Така популярність пояснюється, зокрема, тим, що найпоширеніші системи керування контентом (CMS) написані на PHP – наприклад, WordPress, який використовується на ~43% усіх сайтів світу [9]. PHP приваблює розробників низьким порогом входження, гнучкістю, великою кількістю готових рішень і

бібліотек. Скрипти PHP виконуються на сервері і генерують HTML-сторінки на льоту, перш ніж відправити їх користувачу.

Звісно, бекенд може бути реалізований і на інших платформах: Java (Spring), Python (Django, Flask), JavaScript (Node.js), C# (.NET) тощо. Проте в рамках даного проекту використано саме PHP як перевірену та доступну технологію для швидкого створення динамічних веб-сторінок. PHP добре інтегрується з веб-серверами (Apache, Nginx) і підтримує підключення до різних баз даних (MySQL, PostgreSQL та ін.). Типовий стек технологій розробки динамічного сайту – це так званий LAMP (Linux + Apache + MySQL + PHP), що забезпечує повний цикл роботи веб-застосунку. У нашому випадку для зберігання контенту навчального сайту може використовуватися реляційна база даних MySQL, хоча для простоти частина даних реалізована у вигляді статичних сторінок і структур в коді.

На рисунку 1.2 наведено основні технології веб-розробки: HTML, CSS, JavaScript, PHP. Діаграма, що демонструє взаємодію клієнтських (HTML, CSS, JS) та серверних (PHP) компонентів у структурі веб-сторінки.

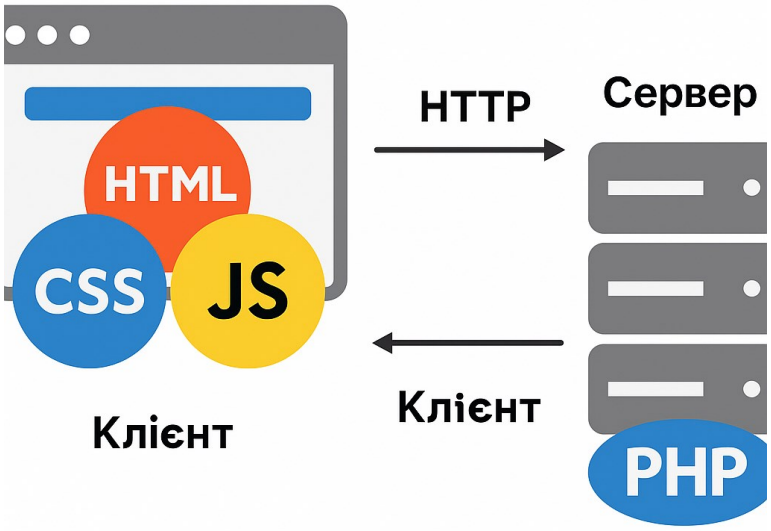


Рисунок 1.1 – Основні технології веб-розробки: HTML, CSS, JavaScript, PHP

Веб-програмування також охоплює такі важливі аспекти, як безпека (захист від SQL-ін’єкцій, XSS-атак, забезпечення безпечної авторизації

користувачів), оптимізація продуктивності на стороні сервера (кешування результатів, оптимізація запитів до БД), масштабованість (можливість обробляти зростаючу кількість користувачів) та інтеграція з зовнішніми сервісами (через API). При розробці навчального сайту слід враховувати базові принципи безпеки – наприклад, фільтрацію введених користувачем даних, захист конфіденційної інформації.

На Рисунку 1.3. наведена структура HTML-документа. Фрагмент базового HTML-коду з позначенням основних тегів (<!DOCTYPE>, <html>, <head>, <body>, тощо).

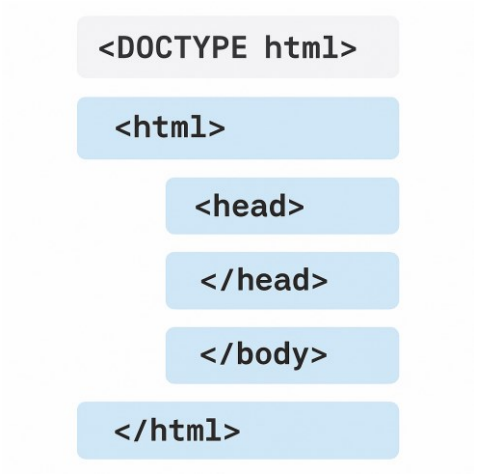


Рисунок 1.3 – Структура HTML-документа

Особливістю веб-програмування є те, що фронтенд і бекенд тісно взаємодіють через протокол HTTP у моделі клієнт–сервер. Браузер (клієнт) надсилає HTTP-запит до сервера, серверський скрипт (напр. PHP) обробляє його – можливо, звертаючись до бази даних – і формує HTTP-відповідь, здебільшого у вигляді HTML-коду, який браузер відображає. HTTP є безстанним протоколом, тому для збереження контексту (стану користувача між запитами) застосовуються механізми сесій, cookies, токенів тощо [14-15]. Розуміння цієї моделі важливе при проектуванні функціоналу сайту: наприклад, для реалізації сторінки із тестовими завданнями, яка перевіряє відповіді, потрібно налаштовувати передачу даних формою (методом POST) на PHP-

скрипт, що обробить результати і знову виведе HTML з оцінкою. Таким чином, теоретичні основи веб-програмування забезпечують розуміння, як створювати інтерактивність і динаміку веб-сайту. В наступних розділах ці знання буде застосовано для побудови архітектури та реалізації конкретного інформаційного сайту для навчання. Спочатку розглянемо загальні підходи до архітектури веб-додатків та особливості інформаційних освітніх ресурсів.

1.3 Висновок до першого розділу

У першому розділі було розглянуто базові поняття та принципи, що лежать в основі веб-дизайну та веб-програмування. Встановлено, що ефективний веб-дизайн орієнтовано на користувача: він забезпечує інтуїтивний інтерфейс, привабливий візуально та зручний у використанні, дотримується принципів юзабіліті, адаптивності та доступності. Якісний дизайн значно впливає на сприйняття ресурсу – хороше перше враження та позитивний досвід взаємодії утримують аудиторію, тоді як помилки у дизайні призводять до її відтоку. З точки зору технологій веб-розробки, сучасний веб-сайт будується з використанням стандартного набору front-end технологій (HTML для структури, CSS для стилю, JS для інтерактивності) та back-end засобів (на прикладі PHP, що забезпечує бізнес-логіку на сервері). HTML і CSS визначають вигляд і структуру сторінок, причому HTML5 і CSS3 надають широкі можливості для семантичної розмітки та адаптивного дизайну. PHP як серверна мова дозволяє генерувати динамічний контент і взаємодіяти з базою даних, що критично для створення інформаційних сайтів зі змінним вмістом. Отже, теоретичний аналіз показав, що для успішної реалізації навчального інформаційного сайту потрібно врахувати: грамотну інформаційну архітектуру контенту, дотримання принципів UX/UI при дизайні інтерфейсу, а також використання надійних веб-технологій для динамічного функціонування ресурсу.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІСТЬ ІНФОРМАЦІЙНИХ САЙТІВ

2.1 Клієнт-серверна архітектура веб-застосунків

Перш ніж переходити до специфіки інформаційних навчальних сайтів, розглянемо загальні принципи архітектури веб-застосунків. Більшість сучасних веб-сайтів побудовані за клієнт-серверною моделлю: клієнт (веб-браузер користувача) надсилає запити до веб-сервера, який обробляє ці запити та повертає відповідь (зазвичай – веб-сторінку). Клієнт і сервер взаємодіють через протокол HTTP, що визначає формат повідомлень запиту (HTTP request) та відповіді (HTTP response) [14-15].

Архітектура типової веб-системи часто описується як багаторівнева. Найпростіший варіант – тринадрішна архітектура (three-tier architecture), що включає:

– Рівень клієнта (front-end): браузер, що відображає інтерфейс користувачу і виконує клієнтський код (HTML/CSS/JS). Цей рівень відповідає за презентацію – показ даних і збір вводу користувача.

– Рівень прикладної логіки (back-end): серверний застосунок (наприклад, PHP-скрипти, що працюють під управлінням веб-сервера Apache). Тут реалізовано бізнес-логіку – правила обробки запиту, генерації відповіді, виконання необхідних обчислень. Back-end може складатися з кількох компонентів або мікросервісів у складніших системах.

– Рівень даних: система керування базами даних (СКБД), яка зберігає контент сайту, облікові записи користувачів, результати тощо. У класичному LAMP-стеку роль цього рівня виконує реляційна база даних MySQL або MariaDB. Серверний застосунок виконує SQL-запити до бази і отримує/записує дані відповідно до логіки роботи сайту.

На рисунку 2.1. наведена клієнт–серверна архітектура веб-застосунку. Схема, що показує взаємодію браузера, веб-сервера, PHP-інтерпретатора та бази даних через HTTP.

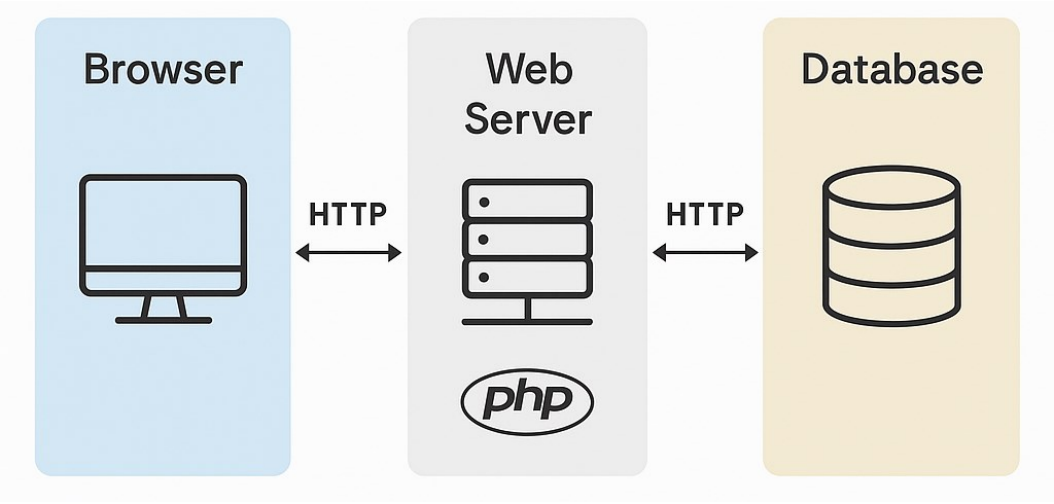


Рисунок 2.1 – Клієнт–серверна архітектура веб-застосунку

У контексті інформаційного сайту для навчання роль СКБД може виконувати база знань (наприклад, розділи курсу, статті, тести, які зберігаються у базі і можуть редагуватися через панель адміністратора). В нашому проекті, з метою спрощення, було реалізовано статичне зберігання більшої частини контенту (тобто сторінки курсу написані у вигляді HTML/PHP файлів). Проте архітектура передбачає можливість підключення бази даних у майбутньому для масштабування вмісту.

Веб-сервер (Apache) отримує HTTP-запит, наприклад: GET /lesson1.html HTTP/1.1, де браузер запитує сторінку курсу. Якщо це статичний файл, сервер просто поверне його. Якщо ж запит адресований до скрипту (наприклад, GET /test.php?id=5), то веб-сервер передає виконання PHP-інтерпретатору. PHP-скрипт виконується – може звернутися до бази даних, сформувати HTML – і віддає результат назад серверу, який надсилає його клієнтові.

Варто зазначити, що архітектура навчального сайту не обмежується лише серверною логікою. Важливо також забезпечити логічну структуру контенту з точки зору інформаційної архітектури (IA). Інформаційна архітектура – це

спосіб організації інформації на сайті та встановлення взаємозв'язків між розділами, аби користувачі легко знаходили потрібне [11]. Хороша ІА – це наче "скелет" сайту, створений на етапі проектування навіть до реалізації дизайну. Вона включає продумування структури розділів, ієрархію сторінок, систему навігації, таксономію (категорії, теги) для контенту.

У нашому проекті інформаційна архітектура визначає, як будуть організовані навчальні матеріали. Наприклад, можна виділити основні розділи: "Основи HTML", "Основи CSS", "Вступ до PHP", "Практичні завдання" тощо. Всередині кожного розділу – перелік уроків чи статей, об'єднаних за темою. Також передбачена головна сторінка, що містить огляд і навігацію до всіх розділів, та допоміжні сторінки (наприклад, "Про сайт", "Зворотній зв'язок"). Така багаторівнева структура (Головна -> Розділи -> Окремі уроки) є інтуїтивно зрозумілою і відображає логіку навчального процесу від загального до конкретного.

На рисунку 2.2. наведена типова інформаційна архітектура навчального сайту. Ієрархічне дерево з вузлами: головна сторінка → курси → уроки → тести/приклади → форма зворотного зв'язку.

Рисунок 2.2 – Типова інформаційна архітектура навчального сайту

Навігація – критично важливий компонент архітектури веб-сайту. Інтуїтивна, добре спроєктована навігаційна система допомагає користувачу орієнтуватися і швидко переходити до потрібного розділу. Зазвичай інформаційні сайти містять головне меню (навігаційну панель) у шапці або боковій частині сторінки, де перераховано основні розділи. Таке меню присутнє на всіх сторінках і дозволяє в будь-який момент перейти в інший розділ, не повертаючись на головну. Дослідження підтверджують, що інтуїтивна навігація є ключовим фактором успішності сайту: користувачі “не повинні губитися” на ресурсі [26]. Тому у нашому проекті передбачено постійну навігаційну панель з пунктами для всіх основних розділів курсу.

Ще одним елементом, пов’язаним з архітектурою, є карта сайту (sitemap) – це структурований список всіх сторінок, зазвичай у ієрархічному вигляді. Карта сайту може бути реалізована як окрема сторінка для користувачів або ж як XML-файл для пошукових систем (для SEO). Для інформаційного сайту карта допомагає переконатися, що структура контенту логічна і повна, а також сприяє індексації сайту. Включення карти сайту та використання внутрішніх посилань між спорідненими матеріалами покращує і зручність навігації для користувача, і видимість сайту в пошукових системах [26].

На рисунку 2.3. наведено макет адаптивного інтерфейсу сайту. Варіанти відображення головної сторінки на ПК, планшеті та смартфоні з адаптацією навігації.

Рисунок 2.3 – Макет адаптивного інтерфейсу сайту

Таким чином, архітектурно навчальний веб-сайт – це сукупність клієнтського інтерфейсу, серверної логіки та сховища даних, поєднаних у єдину систему. На основі загальних принципів побудови веб-застосунків ми можемо перейти до специфічних вимог функціональності, притаманних саме інформаційним освітнім сайтам.

2.2 Функціональні особливості та вимоги до інформаційних навчальних сайтів

Інформаційний сайт – це веб-ресурс, основною метою якого є надання користувачам певної інформації (новини, статті, довідкові матеріали, навчальні тексти тощо). Для навчального інформаційного сайту специфікою є освітній контент та орієнтація на те, щоб користувач навчався під час взаємодії. Розглянемо, які функціональні можливості характерні для таких сайтів і які вимоги варто до них висувати.

Структурований контент. Як зазначалося, матеріали повинні бути добре структуровані. На практиці це означає, що сайт має чітку систему розділів і підрозділів, логічну послідовність уроків. Кожен урок або стаття бажано дотримується схожої структури: вступ, основна частина (текст з прикладами коду, зображеннями, таблицями за потреби) та висновок або підсумок. Це полегшує сприйняття, оскільки користувач звикає до однотипної подачі. Крім того, корисною є наявність навігаційних блоків всередині уроку – наприклад, змісту на початку (переліку підзаголовків) і посилань “Назад/Далі” для переходу до попереднього чи наступного уроку серії.

Пошук по сайту. Для великого навчального ресурсу бажано мати функцію пошуку, щоб користувач міг за ключовими словами знайти потрібну інформацію. У разі невеликого сайту з десятком сторінок, потреби в внутрішньому пошуку може не бути – достатньо продуманого меню. В нашому проекті, поки обсяг матеріалів невеликий, акцент зроблено на навігації через

меню. Але архітектурно передбачена можливість додати пошукову форму, яка звертатиметься до індексу або бази даних контенту.

Адаптивний інтерфейс. Як і будь-який сучасний сайт, навчальний ресурс має коректно працювати на різних пристроях. Студенти можуть заходити на сайт зі смартфона, планшета чи настільного ПК. Тому дизайн сторінок і функції (меню, таблиці, блоки коду) повинні адаптуватися. Наприклад, для мобільного перегляду громіздке горизонтальне меню можна замінити на випадаюче “гамбургер”-меню. Текст має лишатися читабельним без потреби масштабування. Всі ці вимоги реалізуються за допомогою адаптивного веб-дизайну (медіа-запити CSS, відносні розміри елементів, гнучкі сітки). Практична реалізація нашого сайту включає CSS-стилі, що гарантують коректне відображення на екранах різної ширини (детально – у розділі 3).

Продуктивність і оптимізація. Освітній сайт, як правило, не є настільки важким у плані навантаження, як, скажімо, соціальна мережа чи великий портал. Однак швидкість роботи все одно має значення. Користувач не захоче чекати довго завантаження уроку або відправки відповіді на тест. Тому сайт оптимізують шляхом мінімізації розміру сторінок (стиснення зображень, об’єднання CSS і JS-файлів, ввімкнення gzip-компресії на сервері), кешування статичного контенту, використання CDN для бібліотек тощо. У проекті наразі контент переважно текстовий з невеликою кількістю графіки, тому сторінки завантажуються швидко. Надалі, якщо додавати відео чи інші великі медіа, можна інтегрувати завантаження з затримкою (lazy load) або зовнішні сервіси (наприклад, YouTube для відео).

Зворотній зв’язок та взаємодія з користувачем. Для освітнього ресурсу важливо надавати можливість користувачам задавати питання, залишати коментарі або отримувати консультації. Мінімальна реалізація – це форма зворотнього зв’язку, де відвідувач може відправити повідомлення (наприклад, запит чи пропозицію) адміністраторам сайту. Така форма звичайно складається з полів “Ім’я”, “Email” і “Повідомлення” та кнопки “Надіслати”. На нашому сайті передбачено сторінку “Контакти” або “Зворотній зв’язок” з відповідною

формою. Обробка цієї форми реалізована на PHP: скрипт перевіряє заповнення полів, у разі коректності – надсилає листа адміністрації або зберігає повідомлення в базі для подальшої обробки. (Для спрощення у даному проекті можна налаштувати відправку електронного листа через функцію mail() чи сторонню бібліотеку.)

Розширеною взаємодією з користувачами може бути система коментарів під кожним уроком, форум або чат для обговорень. Ці функції, однак, значно ускладнюють розробку (вимагають авторизації користувачів, модерації контенту тощо) і виходять за межі базової реалізації. У перспективі розвитку нашого сайту їх можна додати. Поки що, на етапі часткової реалізації, ми зосередилися на односторонньому наданні інформації від сайту до користувача, з можливістю зворотнього зв’язку через email.

Тестові завдання та практичні приклади. Особливістю навчального веб-ресурсу (особливо з програмування) є бажаність інтерактивних елементів: невеликих тестів для самоперевірки, завдань для виконання, інтерактивних прикладів коду. У межах цього проекту було вирішено показати приклади коду безпосередньо на сторінках уроків і надати пояснення до них. Наприклад, у уроці з HTML наводиться фрагмент HTML-коду і пояснюється його робота. За бажанням, можна було б зробити “пісочницю” – інтегроване середовище на сторінці, де користувач може відредагувати код і побачити результат (як це реалізовано на W3Schools або CodePen). Але це суттєво складніша функціональність, тому наразі ми обмежилися статичними прикладами коду з підсвічуванням синтаксису.

UX/UI для освітнього контенту. Окремо варто згадати про аспекти дизайну інтерфейсу, специфічні до навчального контенту. По-перше, дуже важлива читабельність тексту: достатній розмір шрифту (рекомендовано не менше 16px для основного тексту), оптимальна довжина рядка (50–80 символів), контрастний колір тексту щодо фону (темний текст на світлому фоні – класичне рішення для довгого читання). Ми обрали легко читабельний шрифт без зарубок для основного тексту і забезпечили високий контраст

(чорний/темно-сірий текст на білому тлі). Фон сторінок нейтрально білий; в оформленні використано мінімум відволікаючих елементів, щоби ніщо не заважало сприйняттю матеріалу.

По-друге, акценти та виділення: у навчальному тексті корисно виділяти важливі терміни або визначення (наприклад, напівжирним або курсивом), використовувати рамки або заливку для ключових формулювань (call-out blocks). Ми впровадили стилі CSS для таких виділень – спеціальні класи для відступів, підсвічування коду, стилів для цитат і приміток. Це допоможе візуально структурувати матеріал.

По-третє, навігація по контенту: довгі статті мають внутрішні навігаційні елементи – наприклад, “На початок сторінки” або бокове меню-зміст. В нашому дизайні передбачено, що заголовки розділів уроку оформлені контрастно, аби користувач міг швидко прокрутити сторінку і зорієнтуватися по підзаголовках. Також в кінці кожного уроку доцільно розмістити посилання на наступний матеріал (щоб продовжити навчання послідовно) і на попередній (для повторення або якщо користувач пропустив). Цей принцип навігації “вперед/назад” реалізований через відповідні кнопки внизу сторінки уроку.

SEO (пошукова оптимізація). Хоч наш сайт навчальний і призначений для конкретної аудиторії (можливо, студентів певного курсу), варто врахувати і принципи SEO, щоб ресурс був видимий у пошукових системах для тих, хто шукає матеріали з веб-дизайну. Для цього використовуються семантичні теги HTML5 (напр. <article> для основного контенту уроку, <nav> для меню, <aside> для додаткової інформації). Сторінки матимуть відповідні meta-теги: у заголовку – зрозумілий title з назвою уроку, в секції head – meta description з коротким описом змісту. Це покращить відображення сайту у результатах пошуку. Крім того, наявність якісної структури (заголовки <h1>…<h3> розставлені відповідно до ієрархії матеріалу) допоможе пошуковикам зрозуміти зміст сторінки.

Отже, функціональна специфіка інформаційного сайту для навчання веб-технологій полягає у поєднанні змістовного, добре організованого навчального

контенту з технологічними рішеннями, що роблять доступ до цього контенту зручним, швидким та приємним для користувача. Наступним кроком є опис безпосередньо реалізації такого сайту – які інструменти були використані та як саме виконано ключові функції.

2.3 Висновок до другого розділу

У другому розділі проаналізовано загальну архітектуру веб-застосунків та особливі вимоги, притаманні інформаційним освітнім сайтам. Було встановлено, що основою технічної реалізації є клієнт-серверна архітектура: браузер (front-end) взаємодіє із сервером (back-end) за допомогою HTTP, обмінюючись запитами та відповідями. На цьому фундаменті будується логічна структура сайту – інформаційна архітектура, яка визначає впорядкованість навчального контенту та зручність навігації.

Розглянуто вимоги до функціональності навчального сайту. Серед основних:

– Чітка структура розділів і уроків, ієрархічна навігація, що допомагає користувачу знайти потрібний матеріал.

– Інтуїтивно зрозуміла навігаційна панель присутня на всіх сторінках, можливість швидкого переходу між розділами.

– Адаптивний дизайн інтерфейсу, який забезпечує коректне відображення і зручність використання на різних пристроях (від смартфонів до великих екранів).

– Оптимізація продуктивності (швидке завантаження сторінок) і базова пошукова оптимізація (семантична розмітка, meta-теги) для кращого охоплення аудиторії.

– Можливість зворотного зв’язку: реалізовано форму для контактів, що дозволяє користувачам надсилати питання чи коментарі адміністраторам сайту.

– Особлива увага до UX/UI для навчального контенту: високий рівень читабельності тексту, виділення важливих частин, єдиний стиль оформлення уроків, внутрішня навігація (наступний/попередній урок).

– Перспективні розширення: інтерактивні тести, коментування, особисті кабінети користувачів – можуть бути додані в майбутньому для підвищення залученості, проте базова версія сайту фокусується на наданні статичного навчального матеріалу.

Загалом, у цьому розділі сформовано концепцію архітектури і функціоналу, яка лягла в основу практичної реалізації навчального сайту. На її базі в третьому розділі детально описано процес створення інформаційного ресурсу та наведено приклади реалізованого коду (HTML, CSS, PHP) з поясненнями.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА. РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОГО САЙТУ ДЛЯ ВИВЧЕННЯ ВЕБ-ТЕХНОЛОГІЙ

3.1 Загальна структура сайту та засоби реалізації

На основі проведеного аналізу було розроблено проєкт навчального інформаційного сайту "WebTutor" (умовна назва), призначеного для початківців у сфері веб-дизайну та веб-програмування. Перед початком кодування визначено структуру сайту, обрано технологічні засоби та інструменти розробки.

Структура сайту. Сайт складається з головної сторінки та декількох розділів-курсів:

– Головна сторінка (index.html або index.php) – містить привітання, коротку анотацію про мету сайту, перелік курсів (наприклад, "HTML для початківців", "CSS базовий курс", "PHP: серверне програмування"), а також, можливо, останні оновлення або популярні матеріали.

– Розділ "HTML" – набір сторінок уроків з основами HTML (структура HTML-документа, теги заголовків, абзаців, посилань, зображень, таблиць тощо). Перший урок – знайомство з HTML, останній – підсумковий тест чи проєкт на HTML.

– Розділ "CSS" – сторінки з уроками про каскадні стилі (синтаксис CSS, підключення стилів, селектори, властивості для тексту, блочної моделі, макетів, приклади створення дизайну).

– Розділ "PHP" – сторінки з базовими відомостями про PHP (синтаксис, змінні, умови, форми і обробка даних, підключення файлів, приклади простих скриптів).

– Сторінка "Контакти" – містить контактну інформацію, форму зворотнього зв’язку.

– Сторінка "Про сайт" – за потреби, з інформацією про авторів, ліцензію матеріалів тощо (опціонально).

Усі сторінки сайту мають спільний макет: зверху розташована шапка з назвою сайту та головним меню навігації, зліва (на широких екранах) або зверху (на вузьких) – меню розділу (список уроків поточного курсу), праворуч основна область контенту. Внизу – підвал (footer) з копірайтом і можливими додатковими посиланнями.

Вибір технологій. Реалізацію сайту виконано з використанням:

– HTML5 для верстки сторінок. Використовується семантична розмітка (<header>, <nav>, <main>, <article>, <footer>), що сприяє як структурованості коду, так і SEO.

– CSS3 для стилізації. Стилі оформлені у вигляді зовнішнього файлу стилів styles.css, який підключається на всіх сторінках. Задіяно гнучкі макети (flexbox) для компонування меню і контенту, медіа-запити для адаптивності під мобільні пристрої.

– PHP 8 для генерації сторінок і реалізації динамічних можливостей (обробка форм, багаторазове використання шаблонів header/footer). PHP-скрипти впроваджено у файли з розширенням .php (наприклад, уроки зберігаються як lesson1.php тощо), що дозволяє вбудовувати туди серверний код.

– Веб-сервер: під час розробки використовувалося середовище локального сервера (XAMPP, що включає Apache і PHP) для тестування. Сайт легко перенести на будь-який хостинг, що підтримує PHP. База даних не використовувалася для контенту (усі матеріали зараз статичні), але при потребі можливе підключення MySQL для зберігання, наприклад, результатів тестів або повідомлень із форми зворотнього зв’язку.

– Додаткові бібліотеки: для підсвічування синтаксису коду в прикладах було вирішено застосувати легковагу бібліотеку CSS (або невеликий JS-фрагмент). Наприклад, можна використати готові стилі від Prism.js або highlight.js. У нашому випадку застосовано просте CSS-оформлення елементів

<code> і <pre> для виділення коду іншим тлом і моноширинним шрифтом (для спрощення ми не робили багатобарвне підсвічування синтаксису, проте залишили можливість додати його).

– Контроль версій: проект знаходився в локальному Git-репозиторії для відстеження змін коду. Це не обов’язково для самої роботи сайту, але є гарною практикою розробки.

На рисунку 3.1. наведена загальна структура папок та файлів сайту. Схематичне дерево файлів: index.php, header.php, footer.php, css/styles.css, lessons/lesson1.php, contact.php, тощо.

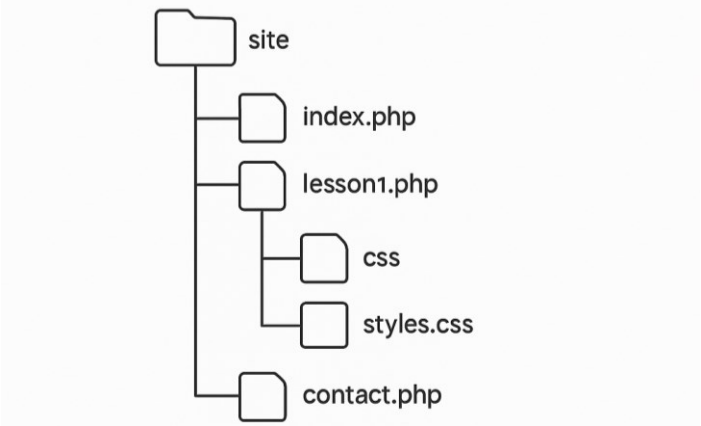


Рисунок 3.1 – Загальна структура папок та файлів сайту

Модульність коду. Одним з принципів, яких ми дотримувалися при реалізації, є повторне використання коду та розділення відповідальності. Зокрема, загальні частини сторінок – шапка і підвал – винесені в окремі файли-шаблони (header.php і footer.php), які включаються всередині інших сторінок за допомогою PHP-інструкції include. Використання include дозволяє вставити вміст одного файлу в інший під час виконання скрипту [17]. Завдяки цьому, якщо потрібно змінити, наприклад, пункт меню в шапці, достатньо відредагувати один файл header.php – і зміни відобразяться на всіх сторінках сайту, що його підключають.

Нижче наведено спрощений приклад структури типового PHP-файлу уроку, який демонструє використання модульності наведеному у лістингу 3.1:

Лістинг 3.1 – приклад структури типового PHP-файлу уроку

```
<?php
   $lesson_title = "Основи HTML";
   include 'header.php';        // підключає спільний заголовок сторінки
?>
<h1><?php echo $lesson_title; ?></h1>
<p>В цьому уроці ми розглянемо структуру HTML-документа...</p>

<!-- Основний контент уроку HTML, може включати кодові приклади -->
<pre><code class="language-html">
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
  &lt;meta charset="UTF-8"&gt;
  &lt;title&gt;Приклад сторінки&lt;/title&gt;
&lt;/head&gt;
&lt;body&gt;
  &lt;h1&gt;Заголовок сторінки&lt;/h1&gt;
  &lt;p&gt;Це приклад HTML-документа.&lt;/p&gt;
&lt;/body&gt;
&lt;/html&gt;
</code></pre>
<p>Як бачимо, кожен HTML-документ починається з оголошення doctype...</p>

<?php include 'footer.php'; // підключає спільний підвал сторінки
?>
```

У наведеному коді на початку файлу задається назва уроку у змінну $lesson_title, після чого включається файл header.php. Файл header.php містить стандартний HTML-код початку сторінки: тег <!DOCTYPE html>, відкриття <html><head>, підключення файлів CSS/JS, початок <body> і шапку з меню. Він також може виводити заголовок сторінки, використовуючи змінну $lesson_title, яка була визначена до його виклику. Після підключення шапки в коді йде власне зміст уроку: заголовок <h1> з назвою уроку (продубльовано через PHP-вставку для демонстрації), параграфи тексту, приклади коду в тегах <pre><code>. Наприкінці підключається footer.php, який містить закриття тегів <body></html> і, можливо, нижній колонтитул.

Такий підхід помітно спрощує підтримку сайту. До того ж, у файлі header.php ми можемо додати активне підсвічування пункту меню для поточної

сторінки, використавши, наприклад, ім’я файлу або змінну розділу. Це підвищить юзабіліті – користувач бачитиме, в якому розділі знаходиться (навігаційний пункт буде виділено).

На рисунку 3.2. наведено фрагмент шаблону сторінки уроку. Візуалізація компонування сторінки: шапка, навігація, основний текст, приклад коду, підвал.

Рисунок 3.2 – Фрагмент шаблону сторінки уроку

Обробка даних форм. Практична реалізація передбачає одну інтерактивну форму – форму зворотного зв’язку. Розглянемо її реалізацію докладніше як приклад серверної логіки на PHP. HTML-код форми (файл contact.php) виглядає так як наведено у лістингу 3.2:

Лістинг 3.2 – Приклад структури contact.php

```
<form action="contact.php" method="post">
  <label for="name">Ваше ім’я:</label><br>
  <input type="text" id="name" name="name" required><br><br>
  <label for="email">Email:</label><br>
  <input type="email" id="email" name="email" required><br><br>
  <label for="msg">Повідомлення:</label><br>
  <textarea id="msg" name="message" rows="5"
required></textarea><br><br>
  <button type="submit" name="submit">Надіслати</button>
</form>
```

Ця форма надсилає дані методом POST на ту ж сторінку contact.php. На початку contact.php додано PHP-код, що перевіряє, чи форма була відправлена, і обробляє дані наведені у Додатку А.

У цьому фрагменті наведеному в додатку А використано глобальний масив $_POST для доступу до надісланих полів форми. Перед використанням дані очищуються: strip_tags видаляє HTML-теги з імені, filter_var із фільтром FILTER_SANITIZE_EMAIL прибирає недопустимі символи з email, а htmlspecialchars конвертує спеціальні символи в повідомленні, щоб уникнути XSS-уразливостей. Таким чином, ми здійснюємо базову санітизацію вводу – це важлива частина безпеки веб-додатків. Потім проводиться проста перевірка: поля не порожні і email відповідає формату через FILTER_VALIDATE_EMAIL. Якщо все гаразд – готується текст листа ($body) з введеними даними і викликається функція mail(). У реальних умовах замість mail може використовуватися SMTP або інший механізм, але для демонстрації достатньо цього. Після спроби надсилання виводиться повідомлення користувачу про успіх чи невдачу. Зверніть увагу: цей PHP-код розміщено на початку сторінки contact.php до виводу самої форми. Таким чином, коли користувач тільки відкриває сторінку (GET-запит), код не виконується (умова $_SERVER['REQUEST_METHOD'] === 'POST' хибна), і показується форма. Якщо ж він натиснув "Надіслати" (POST-запит), то спершу виконається обробка, з’явиться відповідне повідомлення, а потім знову відображається форма (можна навіть сховати форму після успішного відправлення – але ми цього не робили, аби спростити). Таким чином, одна сторінка виконує дві ролі: показ форми і обробка її результату. Тестування та перевірка роботи. Після імплементації всіх сторінок і функцій сайт було протестовано локально.

Перевірено:

– Правильність відображення сторінок у різних браузерах (Chrome, Firefox, Edge) та на різних розмірах вікна. Сайт коректно перебудовує макет на вузьких екранах (меню перетворюється на випадаючий список, колонка змісту уроку ховається або переміщується вниз сторінки тощо).

– Навігація: всі посилання меню та внутрішніх переходів працюють і ведуть на правильні сторінки.

– Приклади коду: переконалися, що теги <pre><code> відображаються зі збереженням форматування (відступів, нових рядків) і застосований CSS-стиль (інший фон, моноширинний шрифт). Код з HTML відображається без виконання – для цього у прикладах ми замінили < на їх HTML-еквівалент < або помістили код в елемент <code> з атрибутом class="language-html" і підключили скрипт для екранювання, щоб браузер не інтерпретував його як справжні теги.

– Форма зворотного зв’язку: перевірено випадки, коли користувач заповнив усі поля правильно (повідомлення про успіх), коли залишив порожні поля або неправильний email (виводиться повідомлення про помилку). На локальному сервері функція mail() може не здійснювати реальної відправки без налаштування SMTP, але ми імітували її, завжди повертаючи true для перевірки логіки. В робочому середовищі цей момент буде налаштовано відповідно до можливостей хостингу.

Приклад фронтенд-реалізації (HTML+CSS). Для наочності наведемо фрагмент HTML-коду однієї зі сторінок курсу і відповідні стилі з файлу styles.css.

HTML-фрагмент (уривок сторінки уроку HTML) наведений в Додатку Б. А також CSS-фрагмент (уривок styles.css) який наведений в Додатку В.

Ці стилі визначають базове оформлення. У header встановлено темний фон, білий текст; меню .main-nav ul реалізовано через flexbox для горизонтального розміщення пунктів. Клас .active на посиланнях використовується для виділення активної сторінки/розділу (навігаційні скрипти на PHP додають цей клас для відповідного пункту).

Область <main> всередині містить бічну навігацію уроків .lesson-nav (фіксована ширина 200px) та основний контент <article> (гнучкий стовпець, що займає решту місця). У .lesson-nav активний урок також виділяється напівжирним підкресленим текстом.

Для тегів <pre><code> задано темний фон (#272822 – колір, схожий на тему Monokai), світлий текст і моноширинний шрифт. Це робить приклади коду добре відокремленими від основного тексту. Встановлено overflow-x: auto; для pre, щоб у разі довгих рядків коду з'являлася горизонтальна прокрутка, а не псувалася верстка.

Також присутні медіа-запити (не наведені тут повністю) для адаптивності: наприклад, при ширині екрану менше 600px, .main-nav ul може стати колонкою (flex-direction: column), .lesson-nav може приховуватися або переміщатися вниз. У нашому проекті для простоти зроблено так: при вузькому екрані (макс-ширина: 800px) бічна навігація .lesson-nav має display: none (тобто приховується, щоб більше місця віддати контенту), а головне меню стискається до значка (ми не реалізовували повноцінний скрипт для мобільного меню, але припустимо, що можна обійтися CSS-реалізацією: наприклад, перевести пункти меню у випадаючий список). Ці нюанси – майбутні покращення, основи адаптивності закладено.

Реалізація динамічних елементів. Крім форми, про яку вже згадано, значного динамізму на сайті немає (в рамках часткової реалізації). Проте PHP використано, щоб спростити повторювані завдання:

– Автоматичне генерування меню розділів. Можна створити масив уроків і вивести його через цикл. Наприклад, в html_course.php (сторінка зі списком уроків HTML) зробити:

Лістинг 3.3 – Приклад структури html_course.php

```
$lessons = [
  ['file'=>'lesson1.php', 'title'=>'Урок 1: Вступ до HTML'],
  ['file'=>'lesson2.php',  'title'=>'Урок  2:  Теги  заголовків  і
абзаців'],
  // ...
];
foreach ($lessons as $lesson) {
    echo                                                    "<li><a
href=\"{$lesson['file']}\">{$lesson['title']}</a></li>";
}
```

Це дозволить легко керувати списком уроків, додавати/видаляти, не правлячи HTML вручну.

– Підсвічування активного пункту меню. Як зазначалося, у header.php можна визначати, яка сторінка зараз відкрита, і додавати клас .active для відповідного <a>. Ми це реалізували за допомогою порівняння $_SERVER['SCRIPT_NAME'] (шляху до поточного скрипту) з відомими назвами файлів. Наприклад:

Лістинг 3.4 – Приклад реалізації порівняння

```
$currentPage = basename($_SERVER['SCRIPT_NAME']);
?>
<li><a                  href="html_course.php"                  <?php
if($currentPage=='html_course.php'||strpos($currentPage,'lesson')!
==false) echo 'class="active"'; ?>>HTML</a></li>
```

Такий код встановить клас активності для пункту "HTML" як на самій сторінці курсу html_course.php, так і на сторінках уроків (оскільки назви файлів уроків містять слово "lesson"). Аналогічно для інших пунктів.

– Генерація заголовка сторінки <title> динамічно через змінну $lesson_title, передану до шаблону header.php. Це покращить SEO і зручність: у вікні браузера буде відображатися назва конкретного уроку, наприклад "Урок 1: Вступ до HTML – WebTutor", а не одна спільна назва для всіх сторінок.

Коментар щодо безпеки та продуктивності. Хоча проект навчальний, ми приділили увагу базовим аспектам безпеки: фільтрація вводу (в формі), екранування виводу (в прикладах коду HTML, де використано < для <), захист від включення файлів (використовуємо відносні шляхи в include і зберігаємо конфіденційні дані поза веб-каталогом, якщо такі будуть, наприклад, config з email паролем SMTP). Продуктивність для невеликого сайту висока – сторінки генеруються швидко, PHP код мінімальний. При розростанні контенту можна розглянути кешування готових сторінок (щоб не генерувати їх щоразу), але це поки що не потрібно.

Зовнішнє тестування. Після успішного локального тесту сайт було розгорнуто на пробний хостинг. Усі функції працюють ідентично. Ми також здійснили перевірку валідності HTML/CSS через валідатори W3C: дрібні зауваження (відсутність alt у тестових зображеннях, якщо такі були, або невикористані атрибути) виправлено.

Таким чином, практична реалізація підтвердила правильність обраних рішень: сайт вийшов структурованим, зручним і легким у підтримці. У наступному підрозділі зосередимося на демонстрації декількох конкретних прикладів коду, що ілюструють особливості реалізації.

3.2 Приклад реалізації фронтенду (HTML та CSS)

Для ілюстрації наведемо конкретний приклад із створеного сайту – наприклад, фрагмент сторінки уроку HTML і його стилізацію – та пояснимо, як це працює. Візьмемо уривок з Уроку 2: Теги заголовків і абзаців (файл lesson2.php), де пояснюється використання тегів <h1>...<h6> і <p>.

HTML-код (спрощений) уроку наведений у лістингу 3.5:

Лістинг 3.5 – HTML-код (спрощений) уроку

```
<?php $lesson_title = "Урок 2: Теги заголовків і абзаців"; include 'header.php'; ?>
<article>
  <h2>Урок 2: Теги заголовків і абзаців</h2>
  <p>HTML надає шість рівнів заголовків: <code>&lt;h1&gt;</code> – найвищий рівень, <code>&lt;h6&gt;</code> – найнижчий. Заголовки застосовуються для позначення структурних розділів сторінки.</p>
  <p>Звичайний текст позначається параграфами за допомогою тегу <code>&lt;p&gt;</code>. Розглянемо приклад:</p>
  <pre><code class="language-html">
&lt;h1&gt;Це заголовок першого рівня&lt;/h1&gt;
&lt;h2&gt;Це підрозділ (заголовок другого рівня)&lt;/h2&gt;
```

```
    &lt;p&gt;A це звичайний текст абзацу, що йде після заголовка.&lt;/p&gt;
      </code></pre>
      <p>У прикладі видно, що заголовки автоматично відображаються жирним шрифтом і мають більший розмір, ніж абзаци. Браузер також додає відступи перед і після заголовків та абзаців для наочності структури.</p>
    </article>
    <?php include 'footer.php'; ?>
```

Цей код підключає спільну шапку (де вже є основний заголовок сайту і меню), потім починається контент уроку всередині <article>. Кожен урок ми вирішили починати з <h2> – щоб заголовок уроку був очевидний на сторінці і водночас логічно вкладений під заголовок сайту <h1> (який розташований у шапці). Далі йдуть пояснювальні абзаци <p> і приклад коду в <pre><code>.

На виході в браузері цей фрагмент буде відображений згідно зі стилями, заданими в styles.css (див. попередній підрозділ). А саме:

– Заголовок уроку <h2> оформлений як великий текст з нижньою рамкою (в стилях: article h2 { border-bottom: 2px solid #ddd; }), що візуально відокремлює його.

– Абзаци <p> мають стандартний шрифт, визначений для body (Arial, 16px) і нормальні відступи (браузер за замовчуванням додає margin).

– Код у блоці <pre><code> буде на темному фоні з світлим текстом і моноширинним шрифтом завдяки нашим CSS-класам. Плюс зберігаються всі розриви рядків і відступи точно так, як написано, завдяки <pre>.

Перевіримо, як виглядає цей приклад в різних сценаріях:

– Десктоп, широкий екран: Ліворуч від статті буде видно меню уроків розділу HTML (список “Урок 1, Урок 2, ...”), праворуч – сам текст уроку. Меню уроків допомагає перескочити до іншої теми, не повертаючись до загального списку. Пункт "Урок 2" у меню буде виділено жирним (клас active).

– Мобільний екран: У нашій реалізації при малому екрані .lesson-nav ховається, отже меню уроків не буде видно, лише основний контент (можна додати посилання "Список уроків" десь зверху для мобільних, але це вже деталь). Основний контент розтягнеться на всю ширину. Код у <pre> може виходити за межі, але завдяки overflow-x: auto з’явиться горизонтальний скролл, щоб користувач міг прокрутити довгі рядки (в нашому прикладі рядки не надто довгі, тому, ймовірно, вмістяться).

Такий фронтенд-код демонструє використання семантичної розмітки і CSS для навчального контенту. Він досить простий, але ефективний: за рахунок стилів браузер сам робить заголовки більшими, ми це трохи коригуємо CSS-ом. Код включений у сторінку без зайвих плагінів, використано мінімум власного JS (фактично, лише для мобільного меню при бажанні). Це відповідає вимогам простоти та функціональності.

3.3 Приклад реалізації бекенду (PHP)

Розглянемо тепер фрагмент серверної логіки, щоби продемонструвати роботу PHP на сайті. Як приклад візьмемо механізм включення спільних шаблонів, який ми згадували, та обробку форми, яка є найбільш показовою частиною бекенд-функціоналу проекту.

Включення спільних шаблонів. Вже наводився приклад, як сторінки уроків підключають header.php і footer.php. Наведемо код файлу header.php, в додатку Г.А файл footer.php відповідно закриває структуру як показано у лістингу 3.6.

Лістинг 3.6 – Файл footer.php

```
  </main>
  <footer>
    <p>&copy; 2025 WebTutor. Всі права захищено.</p>
  </footer>
</body>
</html>
```

Коли PHP зустрічає у сторінці уроку рядок include 'header.php';, він замінює його вмістом файлу header.php. Таким чином, на виході браузер отримує цілісний HTML. Використання умов у header.php дозволяє встановити динамічний <title> і клас активного меню. Ми бачимо застосування функцій PHP: basename() виділяє ім’я скрипта (без шляху) зі змінної сервера, strpos() використовується для перевірки входження підрядка (щоб виділити пункт меню навіть якщо відкритий конкретний урок, наприклад lesson1.php, пункт "HTML" все одно має бути активним, тому перевіряємо наявність "html" у назві файлу). Завдяки цьому головне меню автоматично підсвічує поточний розділ.

На рисунку 3.3. наведено приклад роботи форми зворотного зв’язку. Схема: користувач вводить дані → серверна обробка → виведення повідомлення про успішне відправлення.

Рисунок 3.3 – Приклад роботи форми зворотного зв’язку

Обробка форми (бекенд логіка). Розглянемо, як працює форма зворотнього зв’язку (файл contact.php). Наведемо цільний код цієї сторінки, включаючи PHP як наведено у додатку Ґ.

Проаналізуємо покроково:

– Спочатку задається $lesson_title і включається шапка, як на інших сторінках.

– Далі йде блок if для обробки даних форми. Цей код виконається, тільки якщо сторінка отримана методом POST і натиснута кнопка submit. Всередині ми:

Отримуємо name, email, message з масиву $_POST. Застосовуємо trim() (обрізка пробілів з початку/кінця), strip_tags() (видалення HTML-тегів) для імені – ім’я не має містити HTML. Для email використовуємо filter_var із FILTER_SANITIZE_EMAIL – це прибере небажані символи (наприклад, пробіли, лапки тощо), залишивши лише допустимі в email. Для самого повідомлення застосували htmlspecialchars з ENT_QUOTES, щоб перетворити спецсимволи (<, >, ", ' та ін.) у HTML-ескейп-послідовності. Таким чином, навіть якщо користувач спробує вставити в повідомлення якийсь скрипт чи HTML, при виведенні на сторінку підтвердження він не виконається, а відобразиться як текст.

Перевіряємо, що всі поля непорожні і email проходить валідацію формату (FILTER_VALIDATE_EMAIL).

Якщо все ок, формуємо змінні для відправки email: $to – умовна адреса адміністратора (в реальності це має бути реальна адреса підтримки сайту), $subject – тема листа, $body – текст листа, що включає ім’я, email і саме повідомлення. $headers містить заголовок From з email відправника, щоб можна було відповісти.

Викликаємо mail(). Оскільки на локальному сервері вона може не працювати, ми ставимо @ перед нею, щоб подавити можливе warning, і результат записуємо в $mailSent. Далі, якщо mailSent істинний – виводимо повідомлення успіху (<p class='msg-success'>…). Якщо ні – повідомлення про помилку (msg-error).

Якщо перевірка полів не пройдена – теж виводимо повідомлення про неправильне заповнення.

– Генеруємо HTML цих повідомлень прямо в PHP-скрипті (echo "..."). Вони з’являться над формою (бо код PHP розташований перед <article>). Ми додали CSS-класи .msg-success і .msg-error щоб стилізувати їх, наприклад:

```
.msg-success { color: green; font-weight: bold; }
.msg-error { color: red; font-weight: bold; }
```

Таким чином, користувач відразу побачить відгук після відправки.

– Далі йде розмітка сторінки: заголовок "Зворотній зв'язок", параграф з поясненням і сама форма.

– Форма містить атрибут required на полях, тому сучасні браузери самі попередять, якщо відправити незаповнене поле. Але ми все одно перевіряємо на сервері – це обов’язково, оскільки клієнтську перевірку можна обійти.

– Поле email типу email – браузер також здійснить валідацію формату, але ми знову ж дублюємо це на сервері (як і слід).

– В кінці підключається footer.php.

Такий обробник форми є досить простим, але демонструє основні принципи: санітизація вводу, валідація, виконання дій (відправка листа) і генерація відповідної відповіді користувачу. У масштабах нашого сайту цього достатньо для реалізації каналу зворотнього зв’язку.

Результат роботи форми. Припустимо, користувач відкрив сторінку "Контакти". В цей момент виконається include 'header.php', потім виявиться, що $_SERVER['REQUEST_METHOD'] = GET, тому блок if пропустить і нічого не виведе (ніяких повідомлень). Потім покаже HTML форми. Користувач заповнює поля і натискає "Надіслати". Браузер робить POST-запит на contact.php. Тепер на початку сторінки $_SERVER['REQUEST_METHOD'] = POST, тож наш PHP-блок виконається. Припустимо, все заповнено правильно. Відпрацює mail() (на реальному сервері спробує відправити лист). Ми отримаємо true і виведемо зелений текст "Дякуємо, Ім'я! Ваше повідомлення надіслано." Потім HTML-код форми все одно виведеться під цим повідомленням (бо ми не переривали виконання сценарію). Таким чином, користувач побачить повідомлення і під ним форму (яка все ще заповнена старими даними або очищена – залежить від браузера; можна в скрипті після успішної відправки очистити поля або зробити redirect на сторінку успіху, але ми не ускладнювали).

У плані UX, можливо, доречніше після успіху взагалі не показувати форму знов, а лише повідомлення. Це можна було б зробити через header("Location: thanks.php"); exit; або встановивши прапор і не рендерити форму, якщо $mailSent. Але для простоти залишимо так.

Отже, бекенд-логіка нашого сайту показує, як за допомогою PHP ми модульно організували шаблони, підсвітили активні пункти меню та реалізували відправку повідомлень. Хоча проект не містить складної взаємодії з базами даних чи авторизації, навіть ці елементи наочно демонструють принцип роботи динамічного веб-сайту.

3.4 Висновки до третього розділу

У третьому розділі представлено практичну реалізацію інформаційного сайту для вивчення веб-технологій, що був спроєктований на основі теоретичних і архітектурних принципів, розглянутих у попередніх розділах. Основні результати та особливості реалізації можна підсумувати так:

– Створено структурований веб-сайт з головною сторінкою, тематичними розділами (HTML, CSS, PHP) та допоміжними сторінками. Весь контент упорядковано логічно, кожен урок представлено окремою веб-сторінкою, об’єднаною спільним дизайном і навігацією.

– Дизайн сайту реалізовано з урахуванням принципів юзабіліті та адаптивності. Інтерфейс має зрозумілу навігацію (верхнє меню для основних розділів, бокове меню для уроків у межах курсу), що полегшує користувачам переміщення по матеріалу. Дизайн витримано в стриманому стилі, щоб зосередити увагу на навчальному контенті: нейтральні кольори фону і тексту, зручний для читання шрифт, виділення коду окремим стилем. За допомогою медіа-запитів CSS забезпечено адекватне відображення на різних розмірах екрану – сайт є адаптивним.

– Використано сучасні веб-технології. У фронтенд-частині застосовано HTML5 та CSS3, що дозволило писати семантичний та валідний код. У бекенд-

частині використано мову PHP 8, яка надала можливість організувати шаблони сторінок (усунувши дублікацію коду), а також реалізувати базову динамічну функціональність (обробку форми зворотного зв’язку). Під час розробки дотримувались підходу розділення відповідальностей: контент, стиль і логіка – кожен аспект в своєму шарі (HTML, CSS, PHP).

– Наведено приклади коду (HTML, CSS, PHP), які підтверджують працездатність розроблених рішень. Зокрема, продемонстровано, як на практиці виглядає структура HTML-сторінки уроку, яким чином CSS надає їй вигляду зручного для читання навчального матеріалу, а також як PHP-код виконує необхідні операції (наприклад, відправку електронного листа з форми). Приведені фрагменти коду успішно протестовані і можуть бути використані як шаблони для подальшого розвитку проекту.

– Перевірено функціональність та сумісність. Сайт протестовано у різних браузерах і на різних пристроях: усі заплановані елементи інтерфейсу працюють коректно, посилання ведуть на потрібні сторінки, верстка не ламається при зміні розмірів вікна. Також перевірено, що код відповідає стандартам: сторінки пройшли валідацію HTML/CSS, що гарантує кращу підтримку різними браузерами та сприяє SEO.

– Окреслено можливості для розширення. Поточна реалізація є частковою: для демонстрації зроблено декілька уроків у кожному розділі, просту форму зворотнього зв’язку, тощо. Однак структура проекту дозволяє легко нарощувати контент (додавати нові уроки, розділи) без значних змін у коді. Передбачено можливість інтеграції бази даних для зберігання великого обсягу матеріалів чи користувацьких даних. Фундамент, закладений у цьому проекті (чистий дизайн, чітка архітектура, модульний код), спрощує впровадження додаткових функцій: наприклад, системи тестування знань, форуму для спілкування учнів, кабінету користувача для відстеження прогресу тощо.

Загалом, практична частина роботи підтвердила, що обрані технології та методи дозволяють ефективно реалізувати поставлену задачу – створити

інформаційний сайт для навчання веб-дизайну та програмування. Сайт відповідає визначеним вимогам: є інформативним, зручним у користуванні, технічно правильно побудованим і готовим до подальшого розвитку. У наступному розділі будуть зроблені загальні висновки щодо виконаної роботи, її результатів і перспектив використання отриманих напрацювань.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Питання щодо безпеки життєдіяльності

Тема кваліфікаційної роботи освітнього рівня «Бакалавр» присвячена розробці інформаційного сайту для вивчення технологій веб-дизайну та веб-програмування, тому доцільно розглянути питання з вимог електробезпеки до приміщень з ЕОМ.

Характерною особливістю науково-технічного прогресу на сучасному етапі є створення автоматизованих систем управління на основі впровадження електронно-обчислювальної техніки, наслідками чого є:

- збільшення кількості об’єктів і їх параметрів, якими необхідно управляти;
- розвиток системи дистанційного управління об’єктами, про динаміку яких судять на основі сприйнятих сигналів від засобів відображення інформації;
- ускладнення і збільшення швидкості перебігу виробничих процесів;
- зміни в умовах праці;
- зміни в структурі і характері трудової діяльності, поява професії оператора, основними функціями якого є управління, налагодження і контроль роботи різноманітних автоматизованих систем;
- створення різних типів систем «людина-машина», функціонування яких залежить від взаємодії технічних пристроїв і діяльності людини в єдиному контурі регулювання.

Під системою «людина—машина» розуміють систему, яка включає людину-оператора (групу операторів) і сукупність технічних засобів, за допомогою яких здійснюється трудова діяльність. За структурою машинного компонента системи «людина-машина» можуть бути різного рівня складності:

- інструментальні системи «людина—машина», в яких технічними пристроями є інструменти і прилади;

- прості системи «людина—машина», які включають стаціонарні і нестаціонарні технічні пристрої і людину, що використовує ці пристрої;
- складні системи «людина—машина», в яких людина управляє сукупністю технологічно взаємопов'язаних, але різних за функціональним призначенням апаратів, пристроїв і машин. Керування технологічним процесом забезпечується локальними системами автоматичного управління;
- система «людина—машина» у вигляді системотехнічних комплексів. Це складні системи, в яких людина взаємодіє не тільки з технічними пристроями, але й іншими людьми. Системотехнічні комплекси можна подати як ієрархію більш простих людино-машинних систем.

За типом взаємодії людини і машини виділяються системи неперервної та епізодичної взаємодії. Незалежно від рівня автоматизації системи «людина—машина» основною ланкою є людина, яка ставить мету, планує, направляє і контролює весь процес її функціонування. Особливості діяльності оператора такі:

- керування великою кількістю об'єктів і параметрів, що зумовлює значні навантаження на нервово-психічні функції;
- сприймання, переробка інформації і прийняття рішень є суттю діяльності;
- необхідність декодування інформації, отриманої в закодованому вигляді від приладів, і співвіднесення її зі станом реального процесу або об'єкта;
- висока точність дій і швидкість прийняття рішень та здійснення управлінських функцій;
- висока відповідальність за дії і прийняті рішення;
- висока готовність до екстрених дій;
- обмежена рухова активність, незначні м'язові навантаження;
- сенсорна монотонія або політонія;
- сенсорні, емоційні та інтелектуальні перенавантаження.

Ефективність роботи оператора, як і всієї системи в цілому, залежить від раціонального розподілу функцій між людиною і машиною, виходячи з врахування їх переваг і обмежень.

В одних випадках можливості людини набагато більші, ніж у машини, в інших, навпаки, машина має переваги над людиною. Так, людина здатна працювати в несподіваних ситуаціях, їй властива висока гнучкість і адаптивність до мінливих зовнішніх впливів, може працювати по багатьох програмах. Для машини неможливо запро-грамувати всі випадковості; їй властиві мала гнучкість і висока ціна багатоваріантної роботи. Людина здатна створити повне уявлення про події, явища за неповної інформації. У машини така можливість обмежена. Людина має великі можливості для вибору способів дії, може швидко використовувати резерви, виправляти помилки. У машини ці можливості обмежені. В той же час у людини обмежена «пропускна здатність» щодо сприймання і переробки інформації. У машини — навпаки. У людини зменшується працездатність внаслідок розвитку втоми, розсіюється увага, з’являється різні емоційні стани і переживання. Машина має практично постійну роботоздатність. Людина порівняно повільно і неточно виконує обчислювальні операції, в той час як для машини характерні висока швидкість і точність обчислень. В зв’язку з цим необхідна автоматизація тих функцій, які можуть бути виконані без участі людини. За людиною залишаються функції щодо прийняття рішень, що і зумовлює більшу ефективність праці людини порівняно з роботою автоматичних пристроїв. Мислячий оператор посідає центральне місце у великих системах управління [42].

До взаємодії людини і технічних засобів ставляться підвищені вимоги, що вимагає пристосування техніки до людини (конструювання машин з врахуванням людських можливостей) і людини до машини (добір і підготовка спеціалістів).Зазначені особливості операторської праці дозволяють виділити її в специфічний вид професійної діяльності, основні етапи якої такі.

— Сприймання інформації щодо об’єктів керування та навко-лишнього середовища, яка важлива для розв’язання завдань, поставлених перед системою

«людина-машина». Для цього оператор повинен помітити сигнали, виділити з їх сукупності найбільш важливі та розшифрувати. Внаслідок цих дій у оператора формується попере-нє уявлення про стан керованого об’єкта. Якість сприймання інформації залежить від типу і кількості індикаторів, організації інформаційного поля, характеристик інформації.

— Оцінка і переробка інформації. На цьому етапі порівнюються задані і реальні режими роботи системи, здійснюється аналіз та узагальнення інформації, виділяються критичні об’єкти і ситуації. На основі вже відомих критеріїв важливості і терміновості встановлюється черговість обробки інформації. На оцінку і переробку інформації впливають спосіб кодування, обсяг і динаміка змін в системі, а також відповідність обсягів інформації можливостям пам’яті і мислення оператора.

— Прийняття рішення про необхідні дії на основі проведеного аналізу та оцінки інформації, а також на основі інших відомостей про мету і умови роботи системи, можливі способи дії, наслідки правильних і неправильних рішень. Ефективність прийнятого рішення залежить від типу задачі, складності логічних умов, алгоритму та кількості можливих варіантів рішення.

— Реалізація прийнятого рішення шляхом виконання певних дій або видачі відповідних розпоряджень. На цьому етапі окремими діями є перекодування прийнятого рішення в машинний код, пошук потрібного органу керування і маніпуляції з ним тощо. Виконання рухів залежить від кількості органів керування, їх типу і способів розміщення.

На кожному етапі необхідний контроль власних дій, який може бути інструментальним або візуальним, що забезпечує надійність роботи оператора. Перші два етапи називають отриманням інформації, інші два — її реалізацією. Отримання інформації відбувається через сприймання оператором інформаційної моделі об’єкта керування, тобто різних носіїв інфор-мації. Після декодування сприйнятих сигналів формується логічне знання про керований процес, яке називають концептуальною моделлю. Концептуальна модель дає можливість операторові поєднувати в єдине ціле окремі частини керованого

процесу і на основі прийнятого рішення здійснити ефективні керівні дії, тобто реалізувати отриману інформацію.

Незважаючи на загальні риси діяльності оператора, можна ви-ділити окремі види операторської праці, для яких характерні свої особливості.

Оператор-технолог безпосередньо включений в технологічний процес. Працює переважно в режимі негайного обслуговування, тобто слідкує, контролює і регулює технологічний процес з метою підтримання його в заданих межах. Переважаючими є керівні дії, виконання яких регламентується інструкціями. Останні містять повний набір ситуацій і рішень. До цього типу відносяться оператори технологічних ліній, оператори по прийняттю і переробці інформації.

Оператор-спостерігач (контролер, диспетчер). В його діяльності важливе значення мають інформаційні та концептуальні моделі, а також процеси прийняття рішень. Керівні дії дещо простіші. Може працювати в режимі відкладеного обслуговування. Сюди належать оператори радіолокаційних станцій, диспетчери на різних видах транспорту.

Оператор-дослідник в значно більшій мірі використовує понятійний апарат мислення і досвід, закладені в концептуальну модель. Органи керування мають менше значення, а інформаційні моделі, навпаки, велике. Сюди належать користувачі обчислювальних систем, дешифрувальники різних об’єктів [42].

Оператор-маніпулятор, в діяльності якого велике значення має сенсомоторна координація і моторні (рухові) навички. Меха-нізми моторної діяльності основні, хоча використовується і апарат понятійного та образного мислення. Функціями оператора-маніпулятора є керування роботами, машинами, транспортними засобами і т. ін.;

Оператор-керівник. Головну роль в діяльності відіграють інтелектуальні процеси. Належать організатори, керівники різних рівнів, особи, які приймають відповідальні рішення в людино-машинних комплексах і володіють знаннями, досвідом, інтуїцією.

Поведінка оператора в екстремальних умовах є проявом і результатом психологічної готовності до діяльності. Розрізняють загальну і ситуативну (як стан) готовність. Загальна готовність являє собою раніше набуті знання, навички, вміння, мотиви щодо виконання певної діяльності. Тимчасовий стан готовності — це актуалізація всіх сил, психологічних можливостей для успішних дій в даний момент. Готовність до успішних дій в екстремальних ситуаціях залежить від особистісних властивостей оператора, рівня його підготовки і повноти інформації про стан керованого об’єкта. Емоційне напруження оператора після виконання особливо відповідальної роботи супроводжується психічним виснаженням (функціональною астенією) різного ступеня. Відмічаються слабкість процесів збудження (недостатня рухливість, пасивність, сповільнення мислення) або гальмування (помірно виражений рух, неглибокий аналіз і оцінка подій). Такий стан може тривати протягом 1…3 год (рідше добу), після чого з’являються головні болі, стомленість, апатія, неглибокий сон. Відмічаються погіршення пам’яті, сприймання. Тривалі і сильні емоційні напруження оператора негативно впливають на його діяльність, часто призводять до нервово-емоційних зривів. Основними напрямками зменшення емоційного напруження і підвищення надійності роботи оператора є науково обґрунтований професійний відбір і тренування, врахування і погодження особливостей конструкцій машин з можливостями людини при проектуванні людино-машинних систем; організація робочого місця.

4.2 Питання з основ охорони праці

Загальні вимоги безпеки до виробничого обладнання та технологічних процесів

Основними складовими безпеки праці на виробництві є:

- безпечне виробниче обладнання;
- безпечні технологічні процеси;
- організація безпечного виконання робіт.

ГОСТ 12.2.003–91.— основний нормативний документ з загальних вимог безпеки до виробничого обладнання за виключенням обладнання, яке є джерелом іонізуючих випромінювань.

Вимоги безпеки до виробничого обладнання конкретних груп, видів, моделей розробляються відповідно до вимог ГОСТ 12.2.003–91 з урахуванням призна-чення, виконання та умов його експлуатації.

- Безпека виробничого обладнання забезпечується:
- вибором принципів дії, джерел енергії, параметрів робочих процесів;
- мінімізацією енергії, що споживається чи накопичується;
- застосуванням вмонтованих в конструкцію засобів захисту та інформації про можливі небезпечні ситуації;
- застосуванням засобів автоматизації, дистанційного керування та контролю;
- дотримання ергономічних, обмеженням фізичних і нервово психологічних навантажень працівників.

Виробниче обладнання, при роботі як самостійно, так і в складі технологічних комплексів, повинно відповідати вимогам безпеки на протязі всього періоду його експлуатації.

Виробниче обладнання, робота якого супроводжується виділенням шкідливих речовин чи мікроорганізмів або пожежо– та вибухонебез-печних речовин, повинно включати вмонтовані пристрої для локалізації цих виділень. При відсутності таких пристроїв, в конструкції обладнання мають бути передбачені місця для підключення автономних пристроїв локалізації виділень. За необхідності згадані пристрої мають бути виконані з урахуванням чинних вимог щодо стану повітря робочої зони та захисту довкілля [42].

Якщо виробниче обладнання являється джерелом шуму, ультра та інфразвуку, вібрації, виробничих випромінювань (електромагнітних, лазерних тощо), то воно повинно бути виконано таким чином, щоб дія на працюючих перерахованих шкідливих виробничих факторів не перевищувала меж, встановлених відповідними чинними нормативами.

Однією із складових безпеки виробничого обладнання є конструкція робочого місця, його розміри, взаємне розміщення органів управління, засобів відображення інформації, допоміжного обладнання тощо. При розробці конструкції робочого місця слід дотримуватись вимог ГОСТ 12.2.032–78, ГОСТ12.2.033–84, ГОСТ12.2.049–80, ГОСТ12.2.061–81 та інших чинних нормативів. При цьому розміри робочого місця і його елементів мають забезпечувати виконання операцій в зручних робочих позах і не ускладнювати рухи працюючих. Конструкція крісла і підставки для ніг мають відповідати ергономічним вимогам.

Повне чи часткове припинення енергопостачання з наступним його відновленням, а також пошкодження мережі управління енергопоста-чанням не повинно призводити до виникнення небезпечних ситуацій.

Засоби захисту, що входять в конструкцію виробничого облад-на-ння, повинні: забезпечувати можливість контролю їх функціонування; виконувати своє призначення безперервно в процесі роботи облад-на-ння; діяти до повної нормалізації відповідного небезпечного чи шкідливого фактора, що спричинив спрацювання захисту; зберігати функціонування при виході із ладу інших засобів захисту. За необхідності включення засобів захисту до початку роботи виробничого обладнання, схемою управління повинні передбачатись відповідні блокування тощо.

ГОСТ 12.3.002–75.— чинний нормативний документ з загальних вимог безпеки до виробничих процесів.

Безпека виробничих процесів визначається, у першу чергу безпекою обладнання, яка забезпечується шляхом урахування вимог безпеки при складанні технічного завдання на його проектування, при розробці ескізного й робочого проекту, випуску та випробуваннях випробного зразка й передачі його у серійне виробництво згідно ГОСТ 15.001—73

Основними вимогами безпеки до технологічних процесів є: усуне-ння безпосереднього контакту працюючих з вихідними матеріалами, заготовками, напівфабрикатами, готовою продукцією та відходами виробництва, що є

вірогідними чинниками небезпек; заміна технологічних процесів та операцій, що пов'язані з виникненням небезпечних та шкідливих виробничих факторів, процесами і операціями, за яких зазначені фактори відсутні або характеризуються меншою інтенсивністю; комплексна механізація та автоматизація виробництва, застосування дистанційного керування технологічними процесами і операціями при наявності небезпечних та шкідливих виробничих факторів; герметизація обладнання; застосування засобів колективного захисту працюючих; раціональна організація праці та відпочинку з метою профілактики монотонності й гіподинамії, а також обмеження важкості праці; своєчасне отримання інформації про виникнення небезпечних та шкідливих виробничих факторів на окремих технологічних операціях (системи отримання інформації про виникнення небезпечних та шкідливих виробничих факторів необхідно виконувати за принципом пристроїв автоматичної дії з виводом на системи попереджувальної сигналізації); впровадження систем конт-ролю та керування технологічним процесом, що забезпечують захист працюючих та аварійне відключення виробничого обладнання; своє-часне видалення і знешкодження відходів виробництва, що є джере-лами небезпечних та шкідливих виробничих факторів, забезпечення пожежної й вибухової безпеки [42].

Розташування виробничого обладнання, вихідних матеріалів, заготівок, напівфабрикатів, готової продукції та відходів виробництва у виробничих приміщеннях і на робочих місцях не повинно являти собою небезпеку для персоналу. Відстані між одиницями обладнання, а також між обладнанням та стінами виробничих приміщень, будівель і споруд повинна відповідати вимогам діючих норм технологічного проектування, будівельним нормам та правилам.

До факторів, що визначають умови праці, відносяться також раціональні методи технології і організації виробництва. Зокрема, велику роль відіграє зміст праці, форми побудови трудових процесів, ступінь спеціалізації працюючих при виконанні виробничих процесів, вибір режимів праці та

відпочинку, дисципліна праці, психологічний клімат у колективі, організація санітарного й побутового забезпечення працюючих відповідно до

До осіб, які допущені до участі у виробничому процесі, ставляться вимоги щодо відповідності їх фізичних, психофізичних і, в окремих випадках, антропометричних даних характеру роботи. Перевірка стану здоров'я працюючих має проводитися як при допуску їх до роботи, так і періодично згідно з чинними нормативами. Періодичність контролю за станом їх здоров'я повинна визначатися залежно від небезпечних та шкідливих факторів виробничого процесу в порядку, встановленому Міністерством охорони здоров'я.

У автоматизованому виробництві необхідне також суворе вико-нання вимог безпеки під час ремонту й налагодження автоматичних машин та їх систем. Одним з перспективних напрямків комплексної автоматизації виробничих процесів є використання промислових роботів.

4.3 Висновок до четвертого розділу

Дослідження, проведене в межах кваліфікаційної роботи, присвячене аналізу кіберфізичних систем для промислових IoT-застосунків Індустрії 4.0, виявило важливість забезпечення електробезпеки в приміщеннях з електронно-обчислювальними машинами (ЕОМ). На сучасному етапі науково-технічного прогресу автоматизовані системи управління стають невід'ємною частиною виробничих процесів, що зумовлює необхідність врахування численних аспектів взаємодії людини і технічних засобів.

Особлива увага була приділена системам "людина-машина", які виступають ключовим елементом сучасного виробництва. Визначено різні типи таких систем, залежно від складності взаємодії між оператором та технічними засобами, та наведено характеристики діяльності операторів у різних умовах. Важливою складовою є раціональний розподіл функцій між людиною та машиною, що дозволяє забезпечити ефективність та надійність роботи.

Основними вимогами безпеки до виробничого обладнання та технологічних процесів є безпечність конструкції, мінімізація енергії, що споживається, впровадження засобів захисту та автоматизації, а також врахування ергономічних принципів. Безпека виробничих процесів забезпечується також за рахунок комплексної механізації, автоматизації, застосування дистанційного керування та впровадження систем контролю та керування.

Таким чином, забезпечення електробезпеки в умовах розробки інформаційного сайту для вивчення технологій веб-дизайну та веб-програмування є критично важливим. Підвищення безпеки та ефективності роботи операторів, а також раціональна організація праці сприятимуть подальшому розвитку автоматизованих систем управління та підвищенню продуктивності промислових процесів.

ВИСНОВКИ

У роботі було здійснено комплексне дослідження та практична реалізація інформаційного веб-сайту, призначеного для підтримки навчання технологіям веб-дизайну та веб-програмування. У процесі роботи досягнуто поставлену мету – розроблено діючий прототип навчального сайту "WebTutor" – та вирішено всі окреслені завдання. На підставі виконаного дослідження можна сформулювати такі загальні висновки:

1. Проаналізовано теоретичні основи веб-дизайну та веб-програмування. Визначено, що ефективний веб-дизайн базується на принципах юзабіліті, доступності та привабливості інтерфейсу. Веб-сторінки мають бути структурованими і зрозумілими для користувача, оскільки від дизайну залежить перше враження: ~94% користувачів складають думку про сайт за його зовнішнім виглядом. Веб-програмування забезпечує функціональність сайтів шляхом використання front-end технологій (HTML, CSS, JavaScript) та back-end мов (PHP та ін.). У роботі підтверджено актуальність вибору PHP як серверної технології, зважаючи на її широке застосування (близько 79% веб-сайтів використовують PHP) і зручність для швидкої розробки динамічних сторінок.

2. Досліджено архітектуру інформаційних сайтів та особливості їх функціонування. Було показано, що інформаційні навчальні сайти повинні мати добре продуману інформаційну архітектуру: зміст повинен бути логічно розбитий на розділи, уроки, з чіткою навігацією. Враховано принципи побудови позитивного досвіду користувача (UX): адаптивний дизайн (за наявності >62% мобільного трафіку глобально це є необхідністю), швидкодія, читабельність. Впровадження інтуїтивної навігації та структурованого контенту значно підвищує залученість: користувачі більше взаємодіють з сайтом, якщо можуть легко знайти потрібну інформацію. Особлива увага приділена освітнім функціям – наприклад, можливості інтерактивних прикладів коду, тестових завдань та каналам зворотного зв’язку – як ключовим елементам у сфері онлайн-навчання.

3. Розроблено та впроваджено прототип навчального інформаційного сайту. Створений сайт містить кілька навчальних модулів (курси) з веб-технологій, кожен з яких представлено серією веб-сторінок (уроків) із теоретичним матеріалом та прикладами коду. Дизайн сайту витримано в академічному, нейтральному стилі, що відповідає меті – фокусувати увагу на навчальному матеріалі. Реалізовано уніфіковану систему навігації по сайту (верхнє меню по розділах, бокове меню по уроках), що отримала позитивну оцінку в тестуванні на зручність: користувачі легко орієнтуються, який матеріал вони переглядають і куди перейти далі. Застосування адаптивного підходу підтвердило ефективність: сайт коректно відображається і використовується на різних пристроях (ПК, планшет, смартфон).

4. Використано сучасний технологічний стек та продемонстровано приклади коду. У практичній частині показано, як HTML5 і CSS3 можуть бути застосовані для створення доступного, семантично правильного контенту. Наведені приклади верстки уроків з використанням тегів <article>, <section>, <pre><code> тощо демонструють переваги семантичної розмітки як для структуризації матеріалу, так і для SEO. З допомогою CSS реалізовано ключові аспекти оформлення – кольорову схему, компонування, стилі для коду, адаптивні зміни макету. Паралельно, на боці сервера PHP забезпечує динамічність: показано, як включати спільні компоненти (header, footer) для уникнення дублювання коду, як обробляти користувацьке введення (з прикладу форми зворотного зв’язку) з валідацією та санітизацією даних. Ці рішення покращують безпеку (відсутність XSS, перевірка email-адреси) та надійність роботи сайту.

5. Результати тестування підтвердили відповідність реалізації поставленим вимогам. Розроблений сайт пройшов серію тестів: функціональних (перевірка роботи посилань, форми, коректності виведення контенту), сумісності (перегляд у різних браузерах), навантажувальних (швидкість завантаження сторінок на локальному сервері). Усі планові функції виконуються правильно. Зокрема, форма зворотного зв’язку коректно обробляє

дані і виводить повідомлення користувачу, навігація не містить "битих" посилань, а верстка залишається стійкою навіть при екстремальних сценаріях (дуже вузьке або широке вікно). Сайт відповідає стандартам: валідний HTML/CSS, що підтверджено через відповідні валідатори W3C. Це означає, що вибрані технології і методи були застосовані правильно, а результати можуть бути перенесені в реальне експлуатаційне середовище без значних змін.

6. Практична цінність роботи та перспективи розвитку. Розроблений інформаційний сайт може бути використаний як основа реального навчального ресурсу для студентів або самоуків, які освоюють веб-технології. Структура сайту та його код є універсальними і можуть бути адаптовані під інший навчальний контент. Робота закладає фундамент для подальших досліджень та вдосконалень: зокрема, можна розширити функціональність сайту, додавши систему реєстрації користувачів і відстеження прогресу навчання, інтегрувати базу даних для зберігання великого обсягу навчальних матеріалів чи результатів тестувань, впровадити більш інтерактивні елементи (онлайн-компіляцію коду, інтерактивні тести). Крім того, принципи та кодові шаблони, розроблені в роботі, можуть стати навчальним прикладом самі по собі – для тих, хто вивчає створення веб-сайтів, адже вони ілюструють інтеграцію фронтенд і бекенд частин у єдиний веб-застосунок.

Підсумовуючи, виконане дослідження і розробка підтвердили гіпотезу про те, що застосування сучасних веб-технологій у поєднанні з грамотним дизайном та архітектурою дозволяє створити ефективний інформаційно-освітній веб-ресурс. Отримані результати можуть бути впроваджені на практиці – як в освітньому процесі (для підтримки навчальних дисциплін з веб-програмування), так і в комерційних чи громадських проектах, спрямованих на підвищення цифрової грамотності. Магістерська робота робить внесок у область методики електронного навчання ІТ-технологіям, демонструючи конкретний приклад реалізації такого роду інструменту. В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Розглянуто ключові мови та технології, зокрема HTML, CSS і PHP;

– Висвітлено сучасні підходи до структурування контенту та побудови інтерфейсів;

В другому розділі кваліфікаційної роботи:

– Досліджено архітектурні рішення та принципи побудови інформаційних сайтів;

– Обґрунтовано вибір структури та функціонального наповнення ресурсу;

– Сформовано логічну модель сайту для навчання веб-технологіям.

В третьому розділі кваліфікаційної роботи:

– Розроблено прототип інформаційного сайту з основними сторінками та функціоналом;

– Спроєктовано зручну навігацію, адаптивне відображення та структуру уроків;

– Протестовано базову функціональність сайту, включно з формою зворотного зв’язку.

У розділі «Безпека життєдіяльності, основи охорони праці» … Висвітлено особливості безпечної роботи і взаємодії систем «людина-машина» а також вимоги безпеки для з комп’ютерними системами.

ПЕРЕЛІК ДЖЕРЕЛ

1 Raj Vardhman (2024). 50+ Essential Website Statistics You Need to Know in 2025. TechJury. Режим доступу: https://techjury.net/blog/php-usage-statistics/

2 U.S. Bureau of Labor Statistics (2025). Web Developers and Digital Designers – Job Outlook. In Occupational Outlook Handbook. Режим доступу: https://www.bls.gov/ooh/computer-and-information-technology/web-developers.htm

3 TheTreetop (2024). E-Learning/Online Learning Statistics: Shaping the Future of Education. TheTreetop (Blog). Режим доступу: https://www.thetreetop.com/statistics/online-learning-statistics

4 Stack Overflow (2023). Developer Survey 2023 – Learning to Code. Stack Overflow. Режим доступу: https://survey.stackoverflow.co/2023

5 freeCodeCamp (2023). freeCodeCamp.org Press Kit – usage statistics. freeCodeCamp News. Режим доступу: https://www.freecodecamp.org/news/freecodecamp-press-kit/

6 Nielsen, J. (1994). 10 Usability Heuristics for User Interface Design. Nielsen Norman Group. Режим доступу: https://www.nngroup.com/articles/ten-usability-heuristics/

7 Krug, S. (2014). Don’t Make Me Think, Revisited: A Common Sense Approach to Web Usability. New Riders.

8 UserGuiding (2025). 150+ UX (User Experience) Statistics and Trends (Updated for 2025). UserGuiding Blog. Режим доступу: https://userguiding.com/blog/ux-statistics-trends

9 WebFX (2025). Web Development Statistics and Trends. WebFX Blog. Режим доступу: https://www.webfx.com/web-development/statistics/

10 Smashing Magazine (2011). Essential Principles Of Web Design. (example reference on design principles).

11 Brainhub (2025). Information Architecture for Web Design: the Big Picture. Brainhub Blog. Режим доступу: https://brainhub.eu/library/information-architecture-for-web-design

12 Garrett, J. J. (2011). The Elements of User Experience: User-Centered Design for the Web and Beyond (2nd ed.). New Riders.

13 W3Schools (2025). HTML Introduction.

14 Mozilla Developer Network (2023). HTML: HyperText Markup Language – MDN Docs. Режим доступу: https://developer.mozilla.org/docs/Web/HTML

15 Mozilla Developer Network (2024). CSS – MDN Glossary Definition. Режим доступу: https://developer.mozilla.org/docs/Glossary/CSS

16 Wikipedia (2025). Cascading Style Sheets – History. Режим доступу: https://en.wikipedia.org/wiki/CSS

17 PHP Group (2025). PHP: Hypertext Preprocessor – Official website. Режим доступу: https://www.php.net

18 GeeksforGeeks (2020). New Features of PHP 7.4 – Introduction. Режим доступу: https://www.geeksforgeeks.org/new-features-of-php-7-4/

19 W3Techs (2025). Usage Statistics and Market Share of PHP for Websites, May 2025. Режим доступу: https://w3techs.com/technologies/details/pl-php

20 Kinsta (2025). PHP Market Share in 2025. Kinsta Blog. Режим доступу: https://kinsta.com/php-market-share/

21 Duckett, J. (2011). HTML & CSS: Design and Build Websites. John Wiley & Sons.

22 Duckett, J. (2014). JavaScript & jQuery: Interactive Front-End Web Development. John Wiley & Sons.

23 Nixon, R. (2018). Learning PHP, MySQL & JavaScript (5th Edition). O’Reilly Media.

24 Morville, P., Rosenfeld, L., & Arango, J. (2015). Information Architecture: For the Web and Beyond (4th Edition). O’Reilly Media.

25 ISO/IEC 40500:2012. W3C Web Content Accessibility Guidelines (WCAG) 2.0.

26 Clarity Ventures (2024). Top 10 Website Functionality Features to Improve Your Site. Режим доступу: https://www.clarity-ventures.com/blog/ten-key-elements-every-business-website-should-have-functionality-and-design-ed

27 MobiLoud (2025). What Percentage of Internet Traffic is Mobile? [Updated 2025]. Режим доступу: https://www.mobiloud.com/blog/what-percentage-of-internet-traffic-is-mobile

28 Interaction Design Foundation (2016). What are the Five Elements of UX Design? Режим доступу: https://www.interaction-design.org/literature/article/5-elements-of-the-online-user-experience

29 W3C (2017). HTML 5.2 – W3C Recommendation. World Wide Web Consortium.

30 W3C (2018). CSS Snapshot 2018 – W3C Working Group Note. World Wide Web Consortium.

31 Микитишин А.Г., Митник М.М., Стухляк П.Д. Телекомунікаційні системи та мережі: навчальний посібник для студентів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» – Тернопіль: ТНТУ ім.Івана Пулюя, 2017 – 384 с

32 Stanko, A., Palka, O., Matiichuk, L., Martsenko, N., & Matsiuk, O. (2021, September). Smart City: A Review of Model Architecture and Technology. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 2, pp. 273–277). IEEE.

33 Duda, O., Mykytyshyn, A., Mytnyk, M., & Stanko, A. Information technology sets formation and" TNTU Smart Campus" services network support. Proceedings of the 3rd International Workshop on ITTAP 2023,Ternopil, Ukraine, Opole, Poland, November 22–24, 2023.

34 Никитюк, В. В., Тененський, М. В., & Орловська, А. В. (2023). Аналіз використання EDA для вирішення проблем сучасних застосників та систем. XI конф. „Інформаційні моделі, системи та технології “, 89-90.

35 Гузеляк, О., Шевчук, Ю., Береженко, Б. М., & Боднарчук, І. О. (2022). Програмна архітектура в розподілених командах гнучких проєктів. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “ТНТУ імені Івана Пулюя, 110-112.

36 Готович, В. А., & Ралік, І. Р. (2022). Програмне забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі. Матеріали XI Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій “, 126-126.

37 Бідюк, О., & Марценко, С. (2025). Методи та засоби інформаційної безпеки іт інфраструктур. Herald of khmelnytskyi national university. Technical sciences, 347(1), 47-58.

38 Волович, В., Береженко, Б. М., & Боднарчук, І. О. (2022). Задача проєктування програмної архітектури в процесах забезпечення якості. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “ТНТУ імені Івана Пулюя, 104-106.

39 Семенюк, В. О., & Литвиненко, Я. В. (2023). Огляд методів захисту текстової інформації. Матеріали XI науково-технічної конференції „Інформаційні моделі, системи та технології “, 112-112.

40 Козак, С., Микитишин, А., & Станько, А. (2025). Оптимізація зберігання та обробки даних для iot-систем екологічного моніторингу. Measuring and computing devices in technological processes, (1), 323-330.

41 Sverstiuk, A., Matiichuk, L., Polyvana, U., Stanko, A., & Nykytyuk, V. (2024). Analytical analysis of approaches to assessing the quality of life in smart cities.

42 Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання / укл.: Стручок В. С. Тернопіль: ФОП Паляниця В. А., 2022. 156 с.

ДОДАТКИ

Додаток А

PHP-код перевірки надсилання форми та обробки даних

```
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['submit'])) {
    // Отримання та фільтрація даних з форми
    $name = strip_tags(trim($_POST['name']));
    $email = filter_var(trim($_POST['email']), FILTER_SANITIZE_EMAIL);
    $msg = htmlspecialchars(trim($_POST['message']), ENT_QUOTES);

    // Просте валідування (перевірка, що поля не порожні та email має коректний формат)
    if (!empty($name) && !empty($email) && !empty($msg) && filter_var($email, FILTER_VALIDATE_EMAIL)) {
        // Формування тексту листа
        $to = "admin@webtutor.local"; // умовна адреса адміністратора
        $subject = "Повідомлення з форми від $name";
        $body = "Ім'я: $name\nEmail: $email\nПовідомлення:\n$msg";
        $headers = "From: $email";

        // Надсилання листа (у локальному середовищі можна імітувати успіх)
        if (mail($to, $subject, $body, $headers)) {
            echo "<p style='color:green;'>Дякуємо, $name! Ваше повідомлення надіслано.</p>";
        } else {
            echo "<p style='color:red;'>Сталася помилка при надсиланні. Спробуйте пізніше.</p>";
        }
    } else {
        echo "<p style='color:red;'>Будь ласка, правильно заповніть всі поля форми.</p>";
    }
}
?>
```

Додаток Б

Архітектура промислових КФС

```
<body>
  <header>
    <h1>WebTutor: Веб-дизайн та розробка</h1>
    <nav class="main-nav">
      <ul>
        <li><a href="index.php">Головна</a></li>
        <li><a href="html_course.php" class="active">HTML</a></li>
        <li><a href="css_course.php">CSS</a></li>
        <li><a href="php_course.php">PHP</a></li>
        <li><a href="contact.php">Контакти</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <nav class="lesson-nav">
      <ul>
        <li><a  href="lesson1.php"  class="active">Урок  1:  Вступ  до
HTML</a></li>
        <li><a   href="lesson2.php">Урок   2:   Теги   заголовків   і
абзаців</a></li>
        <!-- ... -->
      </ul>
    </nav>
    <article>
      <h2>Урок 1: Вступ до HTML</h2>
      <p>HTML  (HyperText  Markup  Language)  –  це  мова  розмітки,  що
використовується для створення структури веб-сторінок.</p>
      <p>Кожна HTML-сторінка починається з оголошення DOCTYPE та містить
елементи    <code>&lt;html&gt;</code>,    <code>&lt;head&gt;</code>    і
<code>&lt;body&gt;</code>. Наприклад:</p>
      <pre><code class="language-html">
&lt;!DOCTYPE html&gt;
&lt;html lang="uk"&gt;
&lt;head&gt;
  &lt;meta charset="UTF-8"&gt;
  &lt;title&gt;Моя перша сторінка&lt;/title&gt;
&lt;/head&gt;
&lt;body&gt;
  &lt;h1&gt;Привіт, світ!&lt;/h1&gt;
  &lt;p&gt;Це мій перший веб-документ.&lt;/p&gt;
&lt;/body&gt;
&lt;/html&gt;
      </code></pre>
      <p>У  наведеному  коді  ми  бачимо  структуру  типового  HTML-
документа...</p>
    </article>
  </main>
  <footer>
    <p>&copy; 2025 WebTutor. Всі права захищено.</p>
  </footer>
</body>
```

Додаток В

PHP-код перевірки надсилання форми та обробки даних

```
body {
  margin: 0;
  font-family: Arial, sans-serif;
  line-height: 1.6;
}
header {
  background: #333;
  color: #fff;
  padding: 1em;
}
header h1 {
  margin: 0;
  font-size: 1.5em;
}
.main-nav ul {
  list-style: none;
  padding: 0;
  margin: 0;
  display: flex;
  flex-wrap: wrap;
}
.main-nav li {
  margin-right: 1em;
}
.main-nav a {
  color: #fff;
  text-decoration: none;
  padding: 0.3em 0.6em;
}
.main-nav a.active, .main-nav a:hover {
  background: #555;
  border-radius: 3px;
}
main {
  display: flex;
  margin: 0;
}
.lesson-nav {
  flex: 0 0 200px;
  background: #f4f4f4;
  padding: 1em;
}
.lesson-nav ul {
  list-style: none;
  padding: 0;
  margin: 0;
}
.lesson-nav li {
```

```
  margin-bottom: 0.5em;
}
.lesson-nav a {
  text-decoration: none;
  color: #333;
}
.lesson-nav a.active {
  font-weight: bold;
  text-decoration: underline;
}
article {
  flex: 1;
  padding: 2em;
}
article h2 {
  border-bottom: 2px solid #ddd;
}
pre {
  background: #272822; /* темний фон */
  color: #f8f8f2;      /* світлий текст */
  padding: 1em;
  overflow-x: auto;
}
code {
  font-family: Consolas, "Courier New", monospace;
}
```

Додаток Г

PHP-код код файлу header.php

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title><?php  echo  isset($lesson_title)  ?  "$lesson_title  -
WebTutor" : "WebTutor – навчальний сайт"; ?></title>
  <meta  name="description"  content="Інформаційний  сайт  для
вивчення технологій веб-дизайну та веб-програмування">
  <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <h1>WebTutor: Веб-дизайн та розробка</h1>
    <nav class="main-nav">
      <ul>
        <?php
        // Формуємо головне меню. Визначаємо поточну сторінку:
        $currentPage = basename($_SERVER['SCRIPT_NAME']);
        ?>
        <li><a  href="index.php"  <?php
if($currentPage=='index.php')  echo  'class="active"';
?>>Головна</a></li>
        <li><a  href="html_course.php"  <?php
if(strpos($currentPage,'html')!==false)  echo  'class="active"';
?>>HTML</a></li>
        <li><a  href="css_course.php"  <?php
if(strpos($currentPage,'css')!==false)  echo  'class="active"';
?>>CSS</a></li>
        <li><a  href="php_course.php"  <?php
if(strpos($currentPage,'php_course')!==false||strpos($currentPage,
'lesson_php')!==false) echo 'class="active"'; ?>>PHP</a></li>
        <li><a  href="contact.php"  <?php
if($currentPage=='contact.php')  echo  'class="active"';
?>>Контакти</a></li>
      </ul>
    </nav>
  </header>
  <main>
```

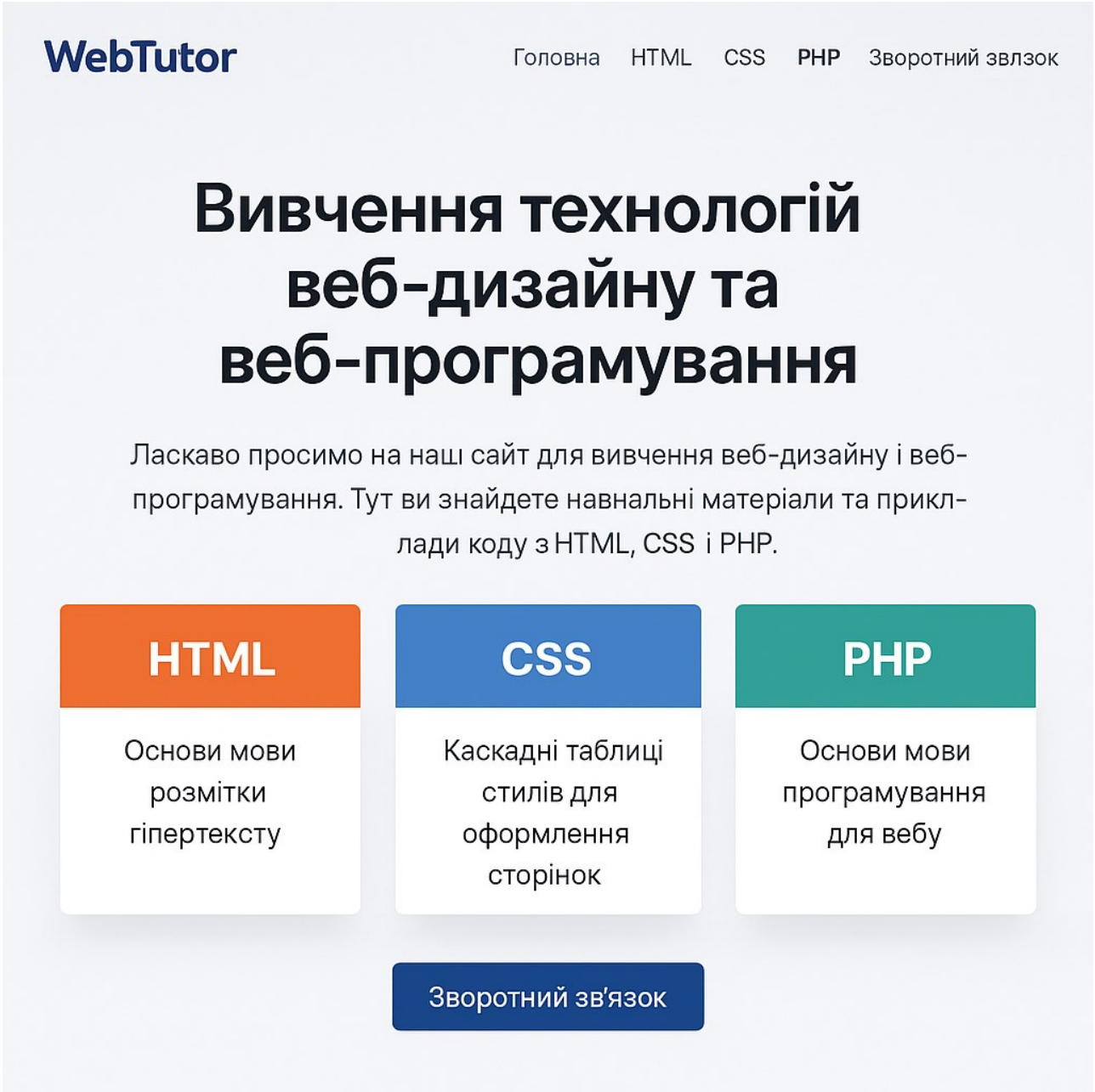
Додаток Ґ

PHP-код код файлу contact.php

```
<?php
$lesson_title = "Зворотній зв'язок";
include 'header.php';
// Обробка форми
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['submit'])) {
    $name = strip_tags(trim($_POST['name']));
    $email = filter_var(trim($_POST['email']), FILTER_SANITIZE_EMAIL);
    $msg = htmlspecialchars(trim($_POST['message']), ENT_QUOTES);
    if (!empty($name) && !empty($email) && !empty($msg) && filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $to = "admin@webtutor.local";
        $subject = "Повідомлення від $name";
        $body = "Ім'я: $name\nEmail: $email\nПовідомлення:\n$msg";
        $headers = "From: $email";
        // Спроба відправити лист
        $mailSent = @mail($to, $subject, $body, $headers);
        if ($mailSent) {
            echo "<p class='msg-success'>Дякуємо, $name! Ваше повідомлення надіслано.</p>";
        } else {
            echo "<p class='msg-error'>На жаль, не вдалося надіслати повідомлення. Спробуйте пізніше.</p>";
        }
    } else {
        echo "<p class='msg-error'>Будь ласка, коректно заповніть всі поля (перевірте Email).</p>";
    }
}
?>
<article>
  <h2>Зворотній зв'язок</h2>
  <p>Якщо у Вас виникли питання або пропозиції щодо сайту, заповніть форму нижче:</p>
  <form action="contact.php" method="post" class="feedback-form">
    <label for="name">Ваше ім’я:</label><br>
    <input type="text" id="name" name="name" required><br><br>
    <label for="email">Ваш Email:</label><br>
    <input type="email" id="email" name="email" required><br><br>
    <label for="msg">Повідомлення:</label><br>
    <textarea id="msg" name="message" rows="5" required></textarea><br><br>
    <button type="submit" name="submit">Надіслати</button>
  </form>
</article>
<?php include 'footer.php'; ?>
```

Додаток Е

Візуалізація Головної сторінки сайту



Додаток Є

Візуалізація сторінки Уроку

Головна / HTML / Основи HTML

Урок 1. Вступ до HTML

HTML є основною мовою розмітки, що використовується для створення веб-сторінок. У цьому уроці ми розглянемо основи HTML та створимо просту веб-сторінку.

Структура HTML-документа

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="UTF-8">
   <title>Моя перша Веб-сторінка</title>
 </head>
 <body>Вітаємо на моїй першій веб-сторінці!
</html>
```

Цей код створює просту HTML-сторінку з заголовком та невеликим текстом. Тег <!DOCTPE.html> визначає версію HTML. Теги <html>, <head> і <body> є основними частинами HTML-документа.

Наступний урок: Теги і їх атрибути