**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

**Практична робота 2**

**з дисципліни «Інтернет-технології та проєктування WEB-застосувань»**

**на тему «РОЗРОБЛЕННЯ АДАПТИВНОГО ІНТЕРФЕЙСУ WEB-ЗАСТОСУНКУ»**

Київ – 2025 року

**ЗМІСТ**

[1 ЗАВДАННЯ ДО ПРАКТИЧНОЇ РОБОТИ 3](#_Toc209501705)

[2 ХІД ВИКОНАННЯ РОБОТИ 5](#_Toc209501706)

[КОНТРОЛЬНІ ПИТАННЯ 9](#_Toc209501707)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 11](#_Toc209501708)

1 ЗАВДАННЯ ДО ПРАКТИЧНОЇ РОБОТИ

Мета: Мета роботи полягає у набутті навичок розроблення адаптивного інтерфейсу веб-застосунку з використанням сучасних засобів верстки, для забезпечення коректного відображення на різних пристроях.

Згідно вимогам стандарту кафедри [1] оформити відповідний розділ «**2 РОЗРОБЛЕННЯ АДАПТИВНОГО ІНТЕРФЕЙСУ WEB-ЗАСТОСУНКУ (ПРАКТИЧНА РОБОТА 2)»**.

У зазначеному розділі необхідно подати *скріншоти, фрагменти коду та текстовий опис реалізації структурних компонентів інтерфейсу та функціонально-стилістичних рішень, передбачених завданням*

Додатково слід навести інформацію про *організацію файлової структури проєкту, особливості розміщення результатів роботи у репозиторії GitHub, приклади логічно обґрунтованих комітів та посилання на відкритий репозиторій*.

**Структурні компоненти інтерфейсу:**

1. Шапка (header) з логотипом та навігаційним меню;
2. Основний блок із розташуванням інформаційних карток та іншого наповнення сайту;
3. Нижній колонтитул (footer), що містить контактну або довідкову інформацію.

**Функціонально-стилістичні вимоги:**

1. Адаптивна навігація, яка трансформується у компактне «бургер-меню» на пристроях із малою роздільною здатністю;
2. Адаптивна сітка контенту, реалізована засобами flexbox або grid, з динамічною перебудовою залежно від ширини екрана (десктоп, планшет, смартфон);
3. Використання медіа-запитів (media queries) для адаптації стилів під різні розміри екранів та орієнтацію пристрою;
4. Застосування відносних одиниць вимірювання (% , em, rem, vw, vh) для забезпечення гнучкості та масштабованості інтерфейсу;
5. Реалізація візуальних ефектів та анімацій, що підвищують зручність і наочність взаємодії з інтерфейсом (плавне відкриття меню, поява контенту, зміна стану елементів при наведенні курсора, тощо).

**Файлова структура веб-застосунку** повинна бути логічно організованою: HTML-документи, стилі, скрипти, графічні та інші ресурси мають зберігатися окремо відповідно до їх призначення, що забезпечує зручність підтримки та масштабування проєкту.

**Вимоги до розміщення результатів роботи в GitHub**. При цьому необхідно дотримуватися таких умов:

1. Використання Git-конвенцій для оформлення репозиторію та комітів згідно з методичними рекомендаціями, наданими за відповідним посиланням [2];
2. Структура репозиторію повинна бути організована із розподілом на відповідні гілки;
3. Коміти мають бути логічно обґрунтованими та змістовно описаними, відображати сутність виконаних змін, уникати надмірної фрагментації або надто загальних описів;
4. Репозиторій повинен містити файл README.md з коротким описом проєкту та інструкцією щодо його запуску;
5. Фінальна версія розробленого коду має бути інтегрована в основну гілку шляхом злиття.

2 ХІД ВИКОНАННЯ РОБОТИ

Хід виконання роботи містить запропонований послідовний план дій, який відображає логіку виконання завдання та охоплює основні етапи розроблення адаптивного інтерфейсу веб-застосунку.

Разом із тим зазначений порядок не є єдино правильним і може бути змінений або деталізований залежно від обраних технічних рішень, індивідуального підходу чи специфіки реалізації.

На початковому етапі роботи необхідно створити кореневу папку майбутнього веб-застосунку, яка міститиме всі елементи проєкту. Усередині цієї папки слід передбачити логічно структуровані підкаталоги, кожен з яких відповідатиме за збереження певного типу файлів. Зокрема:

* /html – для HTML-документів;
* /css – для таблиць стилів;
* /js – для скриптів;
* /assets – для графічних ресурсів, додаткових медіа або статичних файлів.

Після створення файлової структури проєкту в кореневій папці необхідно створити основний HTML-документ під назвою ***index.html,*** який буде головною сторінкою веб-застосунку. Саме цей файл за замовчуванням відкривається браузером або веб-сервером під час завантаження сайту.

У документі слід визначити основні структурні компоненти інтерфейсу. У верхній частині сторінки розміщується шапка (*header*), що містить логотип та навігаційне меню для переходу між розділами. Центральна частина представлена основним блоком (*main*), який використовується для відображення інформаційних карток та іншого контенту. У нижній частині сторінки передбачено колонтитул (*footer*), де подається контактна або довідкова інформація, а також можуть розміщуватися додаткові посилання.

На даному етапі необхідно створити окремий файл таблиць стилів ***style.css***, який буде підключено до головного документа index.html. У цьому файлі слід визначити загальні параметри зовнішнього вигляду сторінки: підібрати та застосувати шрифти, кольорову палітру, відступи між елементами, а також задати початкове позиціонування основних структурних блоків інтерфейсу (*header, main, footer*). Для забезпечення гнучкості та масштабованості інтерфейсу варто використовувати відносні одиниці вимірювання, такі як відсотки (%), em, rem, vw та vh.

Для відображення інформаційних карток та інших елементів основного блоку необхідно реалізувати ***адаптивну сітку***, яка забезпечуватиме зручне та гармонійне розташування контенту на різних пристроях.

Для цього доцільно застосувати сучасні інструменти ***CSS — технології flexbox або grid***, що дозволяють створювати гнучкі багаторядкові структури з рівномірним розподілом простору між елементами.

Важливо передбачити динамічну перебудову основного контенту, наприклад залежно від ширини екрана: на екранах з великою роздільною здатністю (десктопах) картки можуть відображатися у три або більше стовпці, на планшетах — у два, а на смартфонах — у один стовпець. Для досягнення такого ефекту слід застосувати ***медіа-запити (media queries),*** які дозволяють адаптувати стилі сторінки під різні ширини екрана (наприклад, ***<1200px, <768px, <480px***). За їх допомогою можна не лише перебудувати сітку, а й змінити розташування меню, розміри шрифтів, відступи та параметри блоків, що гарантує коректне відображення інтерфейсу на всіх типах пристроїв.

Після створення основної структури сторінки необхідно додати ***навігаційне меню,*** яке забезпечуватиме перехід між основними розділами веб-застосунку. На широких екранах меню реалізується у вигляді горизонтальної панелі з відповідними пунктами. Для пристроїв із малою роздільною здатністю передбачено трансформацію меню у компактне «***бургер-меню»,*** яке відкривається та закривається плавно за допомогою CSS-анімацій або JavaScript. Додатково слід реалізувати візуальні ефекти, що підвищують інтерактивність інтерфейсу: поступову появу карток при завантаженні сторінки та зміну стилю елементів під час наведення курсора.

Після завершення етапів розроблення необхідно здійснити ***перевірку коректності відображення веб-сторінки на пристроях*** із різною роздільною здатністю. У першу чергу сторінку слід переглянути у браузері при зміні ширини вікна — від великого формату (десктоп) до середнього (планшет) та малого (смартфон). Важливо переконатися, що всі структурні компоненти інтерфейсу — шапка, меню, основний блок та колонтитул — залишаються доступними та відображаються коректно, без «зламаних» стилів чи перекривань елементів.

Для імітації роботи сторінки на різних пристроях використовуються інструменти розробника браузера. У більшості сучасних браузерів (Google Chrome, Microsoft Edge, Mozilla Firefox) вони відкриваються комбінацією клавіш F12 або Ctrl+Shift+I. У вікні інструментів необхідно перейти до режиму Device Toolbar (іконка смартфона і планшета у верхній панелі), комбінацією клавіш Ctrl+Shift+M , який дозволяє змінювати роздільну здатність екрана та тестувати сторінку на різних пристроях. За допомогою цього режиму можна перевірити адаптивність меню, сітки контенту, коректність роботи анімацій та візуальних ефектів.

***Робота з Git та розміщення проєкту в GitHub***:

* Ініціалізувати локальний репозиторій Git;
* Створити основну гілку main та робочу гілку develop;
* Для реалізації окремих елементів (наприклад, «feature/header», «feature/footer») створювати тематичні гілки;
* Виконувати коміти з обґрунтованими повідомленнями, які чітко описують сутність змін (наприклад: «Додано адаптивне бургер-меню»);
* Дотримуватись конвенцій оформлення комітів і структури репозиторію згідно з методичними рекомендаціями [2];
* Оформити файл README.md з описом проєкту та інструкцією щодо запуску;
* Після завершення роботи та перевірки коректності – змерджити фінальну версію коду в основну гілку main;
* Надати посилання на репозиторій у звіті.

На завершальному етапі роботи необхідно підготувати звіт. У звіт слід включити фрагменти коду, що демонструють реалізацію структурних компонентів інтерфейсу (header, main, footer) та функціонально-стилістичних рішень (адаптивна навігація, медіа-запити, сітка контенту, анімації). До коду необхідно додати скріншоти веб-сторінки, які ілюструють вигляд інтерфейсу на різних пристроях, а також пояснення щодо застосованих підходів і технологій. Окремо слід подати приклади організації файлової структури проєкту та структури репозиторію, що підтверджує правильність роботи з системою контролю версій. У звіті потрібно зазначити посилання на відкритий репозиторій GitHub із результатами виконаної роботи та навести приклади логічно обґрунтованих комітів. У підсумкових висновках необхідно описати набуті навички, зробити узагальнення отриманих результатів та відзначити можливі напрями удосконалення розробленого інтерфейсу.

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке семантична верстка і які її переваги?
2. Які структурні теги HTML використовуються для побудови інтерфейсу веб-застосунку?
3. Яке призначення тегів <header>, <main> і <footer>?
4. Що таке каскадні таблиці стилів (CSS) і як вони підключаються до HTML-документа?
5. У чому різниця між абсолютними та відносними одиницями вимірювання в CSS?
6. Що таке адаптивний веб-дизайн і чим він відрізняється від респонсивного?
7. Що таке медіа-запити (media queries) і для чого вони застосовуються?
8. Як реалізувати адаптивну сітку за допомогою flexbox?
9. Які можливості надає CSS Grid для організації контенту?
10. Які правила доцільно застосовувати для екранів шириною 1200px, 768px, 480px?
11. Як перевірити правильність відображення інтерфейсу на різних пристроях?
12. Які основні підходи існують для створення «бургер-меню»?
13. Які CSS-властивості використовуються для створення анімацій?
14. У чому різниця між transition і animation у CSS?
15. Як реалізувати плавне відкриття меню?
16. Які способи можна використати для поступової появи контенту при завантаженні сторінки?
17. Як змінюється стиль елементів при наведенні курсора (hover-ефекти)?
18. Які переваги використання анімацій з точки зору UX?
19. Якою має бути структура веб-застосунку для зручності підтримки?
20. У якій папці зазвичай розташовується головний файл index.html?
21. Яке призначення папок /css, /js, /img, /assets у структурі проєкту?
22. Чому важливо відокремлювати HTML, CSS та JavaScript у різні файли?
23. Які труднощі можуть виникнути при реалізації адаптивного меню?
24. Як перевірити коректність роботи анімацій у різних браузерах?
25. Чому важливо протестувати інтерфейс на смартфонах і планшетах?
26. Які є переваги та недоліки використання flexbox у порівнянні з grid?
27. У яких випадках доцільніше застосовувати відсоткові одиниці (%) замість фіксованих px?
28. Як забезпечити масштабованість веб-застосунку під час його подальшої розробки?
29. Які ризики виникають у разі відсутності логічно організованої файлової структури?
30. Чим загрожує використання неконкретних повідомлень комітів (наприклад, «update»)?

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Оформлення текстових документів у навчальному процесі. Стандарт організації (кафедри) СОУ ІСТ 01-22. Для студентів кафедри інформаційних систем та технологій.
2. Conventional Commits URL: <https://www.conventionalcommits.org/en/v1.0.0/#commit-message-with-description-and-breaking-change-footer> (дата звернення 08.08.2025).
3. HTML Підручник. W3Schools українською. URL: <https://w3schoolsua.github.io/html/index.html> (дата звернення: 08.08.2025).
4. HTML. Мова для створення веб-сторінок. W3Schools українською. URL: <https://w3schoolsua.github.io/index.html#gsc.tab=0> (дата звернення: 08.08.2025).
5. HTML Довідник тегів. URL: https://w3schoolsua.github.io/tags/index.html (дата звернення: 08.08.2025).
6. CSS Підручник. W3Schools українською. URL: <https://w3schoolsua.github.io/css/index.html#gsc.tab=0> (дата звернення: 08.08.2025).
7. HTML. Language for creating web pages. W3Schools (English). URL: https://w3schoolsua.github.io/index\_en.html#gsc.tab=0 (дата звернення: 08.08.2025).
8. HTML Tutorial. W3Schools (English). URL: <https://w3schoolsua.github.io/html/index_en.html#gsc.tab=0> (дата звернення: 08.08.2025).
9. Підручники HTML та CSS. HTMLBook. URL: <https://htmlbook.at.ua/news/tutorial_html/1-0-1> (дата звернення: 08.08.2025).
10. Learn to Code. With the world’s largest web developer site. W3Schools. URL: <https://www.w3schools.com/> (дата звернення: 08.08.2025).
11. Український веб-довідник. URL: https://css.in.ua/ (дата звернення: 08.08.2025).
12. Довідник по HTML тегам. URL: <https://css.in.ua/html/tags> (дата звернення: 08.08.2025).
13. Довідник HTML тегів. HTML-CSS.co.ua. URL: <https://html-css.co.ua/dovidnuk-html-tags/> (дата звернення: 08.08.2025).
14. HTML конструювання. HTMLBook. URL: <http://htmlbook.in.ua/>; <http://htmlbook.in.ua/tegs-html/>; <http://htmlbook.in.ua/pr01-html/> (дата звернення: 08.08.2025).
15. CSS-TRICKS. Guides. Articles. URL: <https://css-tricks.com/> (дата звернення: 08.08.2025).