

**ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**  
**КАФЕДРА ІНФОРМАТИКИ І КОМП'ЮТЕРНИХ НАУК**  
**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ**

**МЕТОДИЧНІ РЕКОМЕНДАЦІЇ**

та контрольні завдання до виконання лабораторних робіт,  
практичних (семінарських) занять

**Розгортання інформаційних систем**

*(Архітектурні методики та архітектура програмних додатків у інформаційних системах)*

|                                  |                                     |
|----------------------------------|-------------------------------------|
| для здобувачів                   | Третього курсу                      |
| підготовки                       | першого (бакалаврського) рівня      |
| галузі знань                     | F (12) Інформаційні технології      |
| спеціальності                    | F3 (122) Комп'ютерні науки          |
| освітньо-професійної<br>програми | Комп'ютерні науки                   |
| факультету                       | Інформаційних технологій та дизайну |

**Хмельницький - 2026 р.**

Методичні рекомендації та контрольні завдання до виконання лабораторних робіт з навчальної дисципліни «Розгортання інформаційних систем» (Архітектурні методики та архітектура програмних додатків у інформаційних системах) для здобувачів вищої освіти першого (бакалаврського) рівня спеціальності F3 (122) Комп'ютерні науки

РОЗРОБНИКИ: Наталя Корніловська к.т.н., доц.

Рецензент: Світлана Вишемирська к.т.н., доц.

Затверджено на засіданні кафедри І та КН

Протокол No 6 від 10.03.2026

В.о. завідуючого кафедрою І та КН к.м.н., доц. \_\_\_\_\_ Світлана Моїсеєнко

Відповідальний за випуск Наталя Корніловська к.т.н., доц.

Узгоджено з навчально-методичним відділом

Реєстраційний номер No \_\_\_\_\_

## ВСТУП

Навчальна дисципліна «Розгортання інформаційних систем» спрямована на формування у студентів практичних компетентностей у сфері розгортання, налаштування та підтримки сучасних веб-застосунків і хмарних сервісів. Сучасний ринок праці у галузі інформаційних технологій висуває підвищені вимоги до фахівців, здатних не лише розробляти програмне забезпечення, але й забезпечувати його стабільну роботу у виробничому середовищі.

Процес розгортання (deployment) - це комплекс дій із перенесення застосунку з середовища розробника на сервери, де він буде доступний кінцевим користувачам. Опанування цього процесу є обов'язковою умовою для підготовки повноцінного IT-фахівця.

Цей комплекс методичних рекомендацій охоплює широке коло тем: від базових концепцій хмарних обчислень і моделей надання послуг (SaaS, PaaS, IaaS) до практичного використання сучасних платформ розгортання - Google Sites, GitHub Pages, Netlify та Amazon Web Services. Студенти опановують технології створення статичних веб-сайтів (HTML5, CSS3, JavaScript), системи контролю версій Git, основи кібербезпеки, роботу з REST API та хмарними сховищами.

Кожна лабораторна робота побудована за єдиною структурою: теоретичні відомості з реальними кейсами - детальне покрокове виконання варіанту 1 - висновки до варіанту 1 - перелік варіантів - корисні посилання. Такий підхід дозволяє студентам системно засвоювати матеріал, одразу закріплюючи теорію практичними діями.

Методичні рекомендації призначені для студентів третього курсу спеціальності 122 «Комп'ютерні науки» денної форми навчання. Для виконання лабораторних робіт достатньо мати базові навички роботи з комп'ютером та браузером.

Після завершення курсу студент матиме власний хмарний сайт-портфоліо, доступний у глобальній мережі Інтернет, досвід роботи з хмарними платформами AWS та практичні навички роботи з реальними інструментами сучасного веб-розробника.

# ЗМІСТ

## ВСТУП

ЛАБОРАТОРНА РОБОТА № 1. Розгортання першого хмарного сайту засобами Google Sites

ЛАБОРАТОРНА РОБОТА № 2. Основи HTML5 та CSS3. Підготовка проєкту для розгортання

ЛАБОРАТОРНА РОБОТА № 3. Публікація персонального сайту-портфоліо на GitHub Pages

ЛАБОРАТОРНА РОБОТА № 4. Розгортання веб-сайту на платформі Netlify.

ЛАБОРАТОРНА РОБОТА № 5. Знайомство з хмарними технологіями. Реєстрація та базова конфігурація AWS

ЛАБОРАТОРНА РОБОТА № 6. Хмарне сховище та спільна робота з документами

ЛАБОРАТОРНА РОБОТА № 7. Розміщення сайту у хмарі. Статичний хостинг на Amazon S3

ЛАБОРАТОРНА РОБОТА №8. Знайомство з REST API. Робота з публічними API у Postman

ЛАБОРАТОРНА РОБОТА № 9. Основи кібербезпеки. Захист облікових записів

ЛАБОРАТОРНА РОБОТА № 10. Хмарне сховище. Управління файлами в Amazon S3

ЛАБОРАТОРНА РОБОТА № 11. Безсерверні обчислення. Перша функція AWS Lambda

ЛАБОРАТОРНА РОБОТА № 12. Командний рядок для хмари. Управління S3 через AWS CLI

# **ЛАБОРАТОРНА РОБОТА № 1. ТЕМА: РОЗГОРТАННЯ ПЕРШОГО ХМАРНОГО САЙТУ ЗАСОБАМИ GOOGLE SITES**

## **Вступ до теми**

Ця лабораторна робота відкриває курс навмисно - вона є найпростішою з усього циклу. Студент одразу бачить живий результат: сайт у мережі Інтернет, без жодного рядка коду. Це демонструє головну ідею дисципліни: розгортати застосунок означає зробити його доступним для інших людей. Хмарна платформа бере технічну складність на себе і надає можливість зосередитися на змісті.

Реальний кейс. Компанія Google Sites щороку допомагає мільйонам малих підприємців, викладачів та студентів по всьому світу безкоштовно публікувати свої матеріали в Інтернеті. У 2023 році понад 10 мільйонів активних Google Sites-сайтів існувало у світі - від університетських сторінок до корпоративних інтранетів. Google Sites є класичним прикладом SaaS: жодного встановлення програм, жодних серверів - лише браузер і зміст.

## **Мета роботи**

На практиці зрозуміти сутність моделі хмарних обчислень SaaS (Software as a Service); розгорнути власний веб-сайт у хмарному середовищі без написання програмного коду; усвідомити різницю між локальним розміщенням сервісу і хмарним; набути першого практичного досвіду роботи з хмарними платформами.

## **Завдання до роботи**

У ході виконання цієї роботи студент реєструється у сервісі Google Sites, створює власний сайт з декількома розділами і сторінками, публікує його у глобальній мережі та перевіряє доступність сайту з різних пристроїв.

## **Теоретичні відомості**

### **Хмарні обчислення та моделі надання послуг**

Хмарні обчислення (cloud computing) - це модель надання обчислювальних ресурсів (серверів, сховищ даних, мереж, програмного забезпечення) через мережу Інтернет за принципом «на вимогу». Користувач отримує доступ до необхідних ресурсів без необхідності купівлі та обслуговування власного обладнання.

Розрізняють три основні моделі надання хмарних послуг:

- IaaS (Infrastructure as a Service - інфраструктура як послуга) - провайдер надає обчислювальну інфраструктуру: віртуальні сервери, мережу, сховища.

Приклад: Amazon EC2, Google Compute Engine.

- PaaS (Platform as a Service - платформа як послуга) - провайдер надає платформу для розробки та розгортання застосунків. Приклад: Heroku, Google App Engine, Netlify.

- SaaS (Software as a Service - програмне забезпечення як послуга) - провайдер надає готове програмне забезпечення через браузер. Приклад: Google Docs, Gmail, Google Sites.

Реальний кейс з практики. Відомий стартап Slack спочатку використовував SaaS-підхід і не потребував у клієнтів нічого, окрім браузера. Це дозволило компанії масштабуватися від нуля до 10 мільйонів щоденних активних користувачів менш ніж за три роки - без жодного встановлення програм на боці клієнта.

### Google Sites як SaaS-платформа

Google Sites - це безкоштовний онлайн-конструктор сайтів від компанії Google. Він є класичним прикладом SaaS: користувач відкриває браузер, заходить на сайт sites.google.com і отримує повнофункціональний редактор для створення веб-сайтів. Жодного встановлення програм не потрібно.

| Без хмари (On-Premise)                  | З хмарою - Google Sites (SaaS)              |
|-----------------------------------------|---------------------------------------------|
| Купівля сервера (від 30 000 грн)        | Жодних витрат                               |
| Встановлення ОС та веб-сервера          | Нічого не встановлювати                     |
| Написання HTML/CSS коду                 | Редагування мишею, як у Word                |
| Налаштування безпеки та резервних копій | Google робить автоматично                   |
| Сайт доступний лише з вашого ПК         | Сайт доступний з будь-якої точки світу 24/7 |
| Час до результату: дні та тижні         | Час до результату: 15-30 хвилин             |

### Піраміда відповідальності у SaaS

У хмарних сервісах відповідальність розподіляється між провайдером і клієнтом. У моделі SaaS клієнт відповідає лише за вміст (контент), тоді як провайдер бере на себе весь технічний стек.

| Рівень стека              | On-Premise          | SaaS (Google Sites)                         |
|---------------------------|---------------------|---------------------------------------------|
| Ваш контент (текст, фото) | Ви                  | Ви (тут закінчується ваша відповідальність) |
| Редактор / інтерфейс      | Ви встановлюєте     | Google забезпечує                           |
| Код сайту (HTML/CSS/JS)   | Ви пишете           | Google забезпечує                           |
| Сервер та хостинг         | Ви купуєте          | Google забезпечує                           |
| Фізичний дата-центр       | Ваш або орендований | Google забезпечує                           |
| Безпека та оновлення      | Ви стежите          | Google забезпечує                           |

## Детальне покрокове виконання - Варіант 1

### Крок 1. Відкриття Google Sites

Відкрийте браузер (Chrome, Firefox або Edge). У адресному рядку введіть: [sites.google.com](https://sites.google.com)

Якщо з'явилася сторінка входу - авторизуйтеся за допомогою вашого облікового запису Google (@gmail.com). Якщо облікового запису немає - зверніться до викладача для реєстрації (займає близько 2 хвилин, є безкоштовною).

Зробіть знімок екрана головної сторінки Google Sites після входу.

### Крок 2. Створення нового сайту

Натисніть велику кнопку «+» або «Новий сайт». Вгорі сторінки знайдіть поле «Без назви», клацніть на нього і введіть назву:

Мій перший хмарний сайт - [Ваше прізвище та ім'я]

Натисніть Enter або клацніть поза полем - назва збережеться.

### Крок 3. Наповнення головної сторінки

Ваш сайт повинен містити чотири обов'язкові розділи:

| Номер | Розділ            | Вміст                                                   |
|-------|-------------------|---------------------------------------------------------|
| 1     | Про мене          | Ваше ім'я, спеціальність (122 КН), 1-2 речення про себе |
| 2     | Мої інтереси      | 3-5 ваших захоплень (ігри, музика, спорт тощо)          |
| 3     | Хмарні технології | Що таке SaaS власними словами; який сервіс використали  |
| 4     | Висновок          | 1-3 речення: що дізналися, що сподобалося в роботі      |

Щоб додати текст: натисніть на порожнє місце сторінки та вводьте текст. Для зміни розміру та стилю шрифту використовуйте панель інструментів зверху. Щоб змінити тему оформлення: натисніть «Теми» у верхньому правому куті та оберіть будь-яку тему зі списку. Зробіть знімок екрана редактора з вашим наповненим сайтом.

#### **Крок 4. Додавання другої сторінки**

У нижній частині редактора знайдіть панель «Сторінки». Натисніть «+» або «Нова сторінка». Назвіть нову сторінку: «Хмарні технології». На цій сторінці опишіть: що таке хмарні обчислення; яку модель ви використали (SaaS); чим Google Sites відрізняється від звичайного сайту на власному комп'ютері. Додайте таблицю порівняння SaaS та On-Premise (Вставити - Таблиця).

#### **Крок 5. Публікація сайту**

Це найважливіший крок - тут відбувається розгортання у хмарі. Натисніть синю кнопку «Опублікувати» у правому верхньому куті. У вікні налаштувань у полі «Веб-адреса» введіть адресу у форматі:

`prizvysche-imya-lab1`

(тільки латинські літери та дефіси, без пробілів та кириличних символів). У розділі «Хто може переглядати» оберіть «Всі користувачі Інтернету». Натисніть «Опублікувати». Після публікації ви отримаєте посилання виду:

`https://sites.google.com/view/prizvysche-imya-lab1`

Відкрийте посилання у новій вкладці - це ваш живий сайт. Скопіюйте посилання - воно знадобиться для звіту. Зробіть знімок екрана вашого опублікованого сайту (видима адреса в рядку браузера). Відкрийте сайт на мобільному телефоні та зробіть ще один знімок.

#### **Крок 6. Перевірка результату**

Перевірте виконання наступних умов:

- сайт відкривається за посиланням `sites.google.com/view/ваша-назва`;
- на сайті є всі чотири обов'язкові розділи;
- є друга сторінка «Хмарні технології»;
- сайт коректно відображається на мобільному телефоні або в іншому браузері;
- посилання скопійовано для звіту;
- зроблено щонайменше чотири знімки екрана.

#### **Висновки до Варіанту 1**

Виконавши цю лабораторну роботу, студент на практиці ознайомився з моделлю хмарних обчислень SaaS, розгорнув власний веб-сайт у хмарному середовищі Google Sites без написання коду та без використання власних серверів. Було продемонстровано, що

хмарна платформа бере на себе весь технічний стек - сервери, безпеку, доступність 24/7 - залишаючи користувачу лише роботу зі змістом сайту. Усвідомлено різницю між On-Premise та SaaS підходами до розгортання.

### **Варіанти завдань**

Варіант 2. Створити сайт-портфоліо для вигаданого стартапу на Google Sites з розділами: головна, команда, послуги, контакти. Опублікувати сайт та надати посилання.

Варіант 3. Створити навчальний сайт з теми «Інформаційні технології» на Google Sites. Додати щонайменше 3 сторінки з теоретичним матеріалом та таблицями.

Варіант 4. Розробити сайт-резюме на Google Sites із секціями: профіль, освіта, навички, досягнення. Додати форму зворотного зв'язку.

Варіант 5. Створити сайт для студентського клубу або гуртка на Google Sites. Описати діяльність, заходи, учасників. Опублікувати та перевірити на мобільному пристрої.

Варіант 6. Розробити інформаційний сайт про обрану спеціальність (122 КН) на Google Sites з розділами: опис спеціальності, перспективи, відомі компанії, поради студентам.

### **Корисні посилання та ресурси**

- Google Sites - офіційна довідка: <https://support.google.com/sites/>
- Amazon Web Services: What is SaaS? - <https://aws.amazon.com/what-is/saas/>
- NIST Definition of Cloud Computing - <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>
- Google Sites Help Center - <https://support.google.com/a/users/answer/9314941>

# **ЛАБОРАТОРНА РОБОТА № 2. ТЕМА: ОСНОВИ HTML5 ТА CSS3. ПІДГОТОВКА ПРОЄКТУ ДЛЯ РОЗГОРТАННЯ**

## **Вступ до теми**

HTML і CSS - це мова, якою написаний кожен сайт у світі. Без розуміння цієї основи неможливо розгортати, налаштовувати або виправляти веб-застосунки. У цій роботі студент самостійно пише код і одразу бачить результат - це формує інтуїцію розробника. Крім того, саме тут готуються файли, які у наступних роботах будуть розгорнуті на реальних платформах.

Реальний кейс. Компанія Netflix повністю переробила свій фронтенд у 2013 році, перейшовши на HTML5 та CSS3. До цього сайт використовував Adobe Flash, що не працював на мобільних пристроях. Після переходу час завантаження зменшився на 40%, а кількість мобільних користувачів зросла на 200% протягом першого року. Цей кейс демонструє, як базові веб-технології безпосередньо впливають на бізнес-результати.

## **Мета роботи**

Ознайомитися з базовими концепціями мови розмітки HTML5 та каскадних таблиць стилів CSS3; навчитися створювати структуровану та стилізовану веб-сторінку; підготувати проєкт до подальшого розгортання на платформах GitHub Pages та Netlify; засвоїти поняття адаптивного дизайну.

## **Завдання до роботи**

У ході виконання цієї роботи студент налаштовує середовище розробки VS Code, створює структуру проєкту веб-сайту, пише базовий HTML та CSS код, реалізує адаптивний дизайн та тестує сайт локально у браузері.

## **Теоретичні відомості**

### **Чому ця тема є у вашій дисципліні**

Ви обрали спеціальність 122 «Комп'ютерні науки». Це означає, що впродовж навчання вам доведеться стикатися з веб-застосунками: розгортати їх на серверах, налаштовувати, виправляти помилки, читати і розуміти чужий код. Щоб це робити впевнено, треба знати основу - HTML і CSS.

Уявіть будівлю. HTML - це цегла, перекриття, стіни: тобто сама конструкція. CSS - це штукатурка, фарба, меблі: тобто зовнішній вигляд. JavaScript (у наступних роботах) - це

електрика та ліфти: те, що рухається і реагує на дії мешканців. Без першого шару решта просто неможлива.

У цій роботі ви власноруч напишете код і одразу побачите результат у браузері. Саме тут ви створюєте файли, які у наступних роботах будуть розгорнуті на реальних хмарних платформах - GitHub Pages та Netlify. Тому ця робота є фундаментом для всіх наступних.

## HTML - мова, яку «читає» браузер

HTML (HyperText Markup Language - мова гіпертекстової розмітки) - це стандартна мова для опису структури веб-сторінок. Важливо одразу зрозуміти: HTML - НЕ мова програмування. У ньому немає умов, циклів, функцій. Він просто описує, що саме знаходиться на сторінці: «тут заголовок», «тут абзац тексту», «тут зображення», «тут посилання».

Коли ви відкриваєте будь-який сайт у браузері, браузер завантажує HTML-файл, читає його зверху вниз і будує з нього те, що ви бачите на екрані. Цей процес називається рендерингом.

## Як влаштований HTML-тег

Весь HTML складається з тегів. Тег - це команда для браузера, записана у кутових дужках. Більшість тегів мають відкриваючий і закриваючий варіант:

```
<p>Це абзац тексту</p>
^--відкриваючий      ^--закриваючий (зі скісною рисою /)

<a href="#about">Про мене</a>
  ^--атрибут (додаткова властивість тегу)
```

Атрибути - це додаткові параметри тегу. Наприклад, у тегу <a> атрибут href вказує, куди веде посилання; у тегу <img> атрибут src вказує шлях до зображення, а alt - підпис для незрячих і для випадку, коли зображення не завантажилось.

## Базова структура HTML5-документа

Кожен HTML-файл повинен починатися з певних обов'язкових рядків. Розберемо кожен:

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Назва вкладки браузера</title>
  <link rel="stylesheet" href="style.css">
</head>
```

```
<body>
  <!-- Весь видимий вміст сторінки -->
  <script src="script.js"></script>
</body>
</html>
```

| Рядок коду                               | Що робить                                      | Чому важливий                                                                            |
|------------------------------------------|------------------------------------------------|------------------------------------------------------------------------------------------|
| <!DOCTYPE html>                          | Повідомляє браузеру: «це сучасний HTML5»       | Без нього браузер переходить у «режим сумісності» і може відображати сторінку некоректно |
| <html lang="uk">                         | Вказує мову вмісту сторінки                    | Потрібно для пошукових систем і програм зчитування для незрячих                          |
| <meta charset="UTF-8">                   | Встановлює кодування символів                  | Без нього українські літери перетворюються на «кракозябри»                               |
| <meta name="viewport" ...>               | Налаштовує відображення на мобільних пристроях | Без нього сайт на смартфоні буде крихітним і нечитабельним                               |
| <title>...</title>                       | Задає текст на вкладці браузера                | Відображається у вкладці та у результатах пошуку Google                                  |
| <link rel="stylesheet" href="style.css"> | Підключає CSS-файл зі стилями                  | Без нього CSS не буде застосований до сторінки                                           |
| <!-- коментар -->                        | Коментар - не відображається у браузері        | Допомагає орієнтуватися у коді                                                           |
| <script src="..."></script>              | Підключає JavaScript-файл                      | Розміщується в кінці <body>, щоб сторінка завантажувалась швидше                         |

## Основні семантичні теги HTML5

У HTML5 з'явилися «семантичні» теги - теги, назва яких описує зміст, а не вигляд. Наприклад, тег <nav> говорить: «тут знаходиться навігаційне меню». Це допомагає браузерам, пошуковим системам і програмам для незрячих краще розуміти структуру сторінки.

| Тег           | Призначення                                                 | Приклад використання              |
|---------------|-------------------------------------------------------------|-----------------------------------|
| <nav>         | Навігаційне меню сайту                                      | <nav class="navbar">...</nav>     |
| <section>     | Логічний розділ сторінки (про мене, проекти...)             | <section id="about">...</section> |
| <h1>—<br><h6> | Заголовки шести рівнів (h1 - найважливіший, h6 - найменший) | <h1>Привіт, я Іван</h1>           |
| <p>           | Абзац тексту                                                | <p>Короткий опис проєкту.</p>     |
| <a>           | Гіперпосилання (anchor - якір)                              | <a href="#about">Про мене</a>     |

| Тег        | Призначення                                | Приклад використання                  |
|------------|--------------------------------------------|---------------------------------------|
| <img>      | Зображення (не має закриваючого тегу)      |   |
| <ul>, <li> | Ненумерований список і його елементи       | <ul><li>Python</li><li>HTML</li></ul> |
| <form>     | Форма для введення даних користувачем      | <form id="contactForm">...</form>     |
| <footer>   | Підвал сторінки (копірайт, посилання)      | <footer>© 2026 Іван Іванов</footer>   |
| <div>      | Універсальний блок-контейнер без семантики | <div class="container">...</div>      |

### CSS - як зробити сторінку красивою

CSS (Cascading Style Sheets - каскадні таблиці стилів) - мова опису зовнішнього вигляду HTML-елементів. Якщо HTML відповідає за структуру (що є на сторінці), то CSS відповідає за оформлення: кольори, шрифти, відступи, розміри, розташування блоків.

Слово «каскадні» означає, що стилі успадковуються і накладаються один на одного: стиль, заданий для <body>, поширюється на всі елементи всередині, якщо не перевизначений точніше.

### Як виглядає CSS-правило

```
селектор {
  властивість: значення; /* коментар */
  властивість: значення;
}

/* Приклад: */
h1 {
  color: #333333;          /* колір тексту - темно-сірий */
  font-size: 2rem;         /* розмір шрифту: 2 × базовий (зазвичай 32px) */
  margin-bottom: 16px;     /* зовнішній відступ знизу - 16 пікселів */
}
```

Розберемо ключові поняття:

| Термін       | Що означає                                            | Приклад                                                                              |
|--------------|-------------------------------------------------------|--------------------------------------------------------------------------------------|
| Селектор     | Вказує, до яких HTML-елементів застосовується правило | h1 - всі заголовки h1; .navbar - елементи з класом navbar; #logo - елемент з id logo |
| Властивість  | Що саме змінюємо у зовнішньому вигляді                | color, font-size, margin, padding, background...                                     |
| Значення     | На яке значення встановлюємо властивість              | red, #333, 16px, 2rem, flex...                                                       |
| px (пікселі) | Фіксована одиниця розміру (1 піксель екрана)          | font-size: 16px                                                                      |

| Термін | Що означає                                                                     | Приклад                     |
|--------|--------------------------------------------------------------------------------|-----------------------------|
| rem    | Розмір відносно базового шрифту (зазвичай 16px). Переважна одиниця для шрифтів | font-size: 2rem = 32px      |
| %      | Відсоток від розміру батьківського елемента                                    | width: 100% - на всю ширину |

## CSS-змінні (Custom Properties)

CSS-змінні - одне з найпотужніших нововведень CSS3. Вони дозволяють зберегти значення (наприклад, основний колір сайту) в одному місці і використовувати в будь-якому місці стилів. Якщо потрібно змінити колір - змінюєте лише одне значення, а не шукаєте його по всьому файлу.

```
:root {
  /* Оголошення змінних: завжди починаються з -- */
  --primary-color: #2c3e50; /* основний темний колір */
  --accent-color: #3498db; /* колір акценту (синій) */
  --font-main: 'Segoe UI', sans-serif;
}

/* Використання змінної: */
nav {
  background-color: var(--primary-color); /* посилаємось на змінну */
}
a:hover {
  color: var(--accent-color);
}
```

Блок `:root` - це псевдоклас, що означає «кореневий елемент документа» (тобто `<html>`). Змінні, оголошені тут, доступні у всьому CSS-файлі.

## Flexbox - сучасна модель розташування елементів

Flexbox (Flexible Box Layout) - сучасний спосіб розташування елементів у рядок або стовпець. Дуже зручний для навігаційних меню, карток, галерей. До Flexbox веб-розробники використовували складні «хаки» з `float` і `position` - тепер це значно простіше.

```
.navbar .container {
  display: flex; /* увімкнути Flexbox */
  justify-content: space-between; /* розподілити простір між елементами */
  align-items: center; /* вирівняти по вертикалі по центру */
}
```

| Властивість     | Значення      | Що робить                                                                       |
|-----------------|---------------|---------------------------------------------------------------------------------|
| display: flex   | -             | Перетворює контейнер на flex-контейнер; дочірні елементи стають flex-елементами |
| justify-content | space-between | Розподіляє елементи з рівними проміжками; крайні - до країв                     |

| Властивість     | Значення | Що робить                                                              |
|-----------------|----------|------------------------------------------------------------------------|
| justify-content | center   | Центрує всі елементи по горизонталі                                    |
| align-items     | center   | Вирівнює елементи по центру по вертикалі                               |
| flex-wrap       | wrap     | Дозволяє елементам переноситися на наступний рядок, якщо не вміщаються |
| gap             | 16px     | Відступ між flex-елементами (зручніше, ніж margin)                     |

## Адаптивний дизайн: чому сайт повинен добре виглядати на телефоні

Адаптивний (responsive) дизайн - підхід до створення сайтів, при якому вигляд сторінки автоматично підлаштовується під розмір екрана: чи то це великий монітор, ноутбук, планшет чи смартфон. Станом на 2024 рік понад 58% веб-трафіку у світі припадає на мобільні пристрої - тому адаптивність є не «бонусом», а обов'язковою вимогою.

Якщо сайт НЕ адаптивний, на телефоні він виглядає так: весь вміст стиснутий і крихітний, щоб побачити текст - треба збільшувати пальцями, навігаційне меню виходить за межі екрана. Такий сайт Google опускає нижче в результатах пошуку.

## Медіа-запити (@media)

Медіа-запити - це умовні блоки CSS, які застосовуються лише тоді, коли виконується певна умова (наприклад, ширина екрана менша за 768px). Усе, що знаходиться всередині фігурних дужок медіа-запиту, перекриває звичайні стилі.

```
/* Це стиль для всіх екранів (за замовчуванням): */
.nav-links {
  display: flex; /* посилання навігації видимі, у рядок */
}
.burger {
  display: none; /* кнопка-«гамбургер» прихована */
}

/* Це стиль тільки для екранів шириною 768px і менше: */
@media (max-width: 768px) {
  .nav-links {
    display: none; /* список посилань ховаємо */
  }
  .burger {
    display: flex; /* кнопку-«гамбургер» показуємо */
  }
}
```

Число 768px - це умовна межа між планшетом і телефоном. Вона не є стандартом, але широко використовується у веб-розробці. Ви можете змінити це число під свій дизайн.

## VS Code - ваше робоче середовище

Visual Studio Code (VS Code) - безкоштовний редактор коду від Microsoft. Це найпопулярніший редактор серед веб-розробників у світі (за опитуванням Stack Overflow 2023 - 73% розробників використовують VS Code). Він підсвічує синтаксис (різні частини коду виділяються різними кольорами), підказує теги і властивості при введенні, виявляє помилки прямо у редакторі.

| Програма                  | Призначення                                                              | Де завантажити                                                                                              |
|---------------------------|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| VS Code                   | Редактор коду з підсвічуванням синтаксису, автодоповненням, розширеннями | <a href="https://code.visualstudio.com">code.visualstudio.com</a>                                           |
| Google Chrome або Firefox | Перегляд та тестування сайту. DevTools (F12) - інструменти розробника    | <a href="https://chrome.google.com">chrome.google.com</a> або <a href="https://firefox.com">firefox.com</a> |

### Необхідні розширення VS Code

Розширення (Extensions) - це додаткові плагіни, що розширюють можливості VS Code. Щоб їх встановити: натисніть **Ctrl+Shift+X** (або кнопку квадратів у лівій панелі) → введіть назву розширення → натисніть **Install**.

| Розширення       | Автор       | Що робить                                                                                         | Навіщо потрібне                                                   |
|------------------|-------------|---------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| Live Server      | Ritwick Dey | Запускає локальний веб-сервер і автоматично оновлює браузер при кожному збереженні файлу (Ctrl+S) | Без нього треба вручну натискати F5 у браузері після кожної зміни |
| Prettier         | Prettier    | Автоматично форматує код при збереженні: вирівнює відступи, розставляє лапки                      | Код стає охайним і читабельним; важливо для командної роботи      |
| HTML CSS Support | ecmel       | Підказує назви CSS-класів прямо у HTML-файлі при введенні атрибута <code>class="..."</code>       | Допомагає уникнути помилок у назвах класів                        |

### Як браузер обробляє HTML і CSS: загальна схема

Розуміння цього процесу допоможе вам краще діагностувати помилки. Ось що відбувається, коли ви відкриваєте сторінку або натискаєте «Go Live» у VS Code:

| Етап                 | Що відбувається                                                                                                   | Де можна побачити           |
|----------------------|-------------------------------------------------------------------------------------------------------------------|-----------------------------|
| 1. Завантаження HTML | Браузер читає <code>index.html</code> зверху вниз, будує DOM (Document Object Model) - дерево всіх елементів      | DevTools → вкладка Elements |
| 2. Завантаження CSS  | Браузер знаходить тег <code>&lt;link&gt;</code> і завантажує <code>style.css</code> . Будує CSSOM - дерево стилів | DevTools → вкладка Styles   |

| Етап          | Що відбувається                                                                               | Де можна побачити           |
|---------------|-----------------------------------------------------------------------------------------------|-----------------------------|
| 3. Рендеринг  | Браузер об'єднує DOM і CSSOM, обчислює розміри і позиції елементів, малює піксель за пікселем | Те, що ви бачите в браузері |
| 4. JavaScript | Якщо є <code>&lt;script&gt;</code> , браузер виконує JavaScript після побудови DOM            | DevTools → вкладка Console  |

Саме тому `<script>` розміщується наприкінці `<body>`: якщо поставити його у `<head>`, JavaScript завантажуватиметься до того, як HTML-елементи з'являться у DOM, і скрипт не знаходить їх.

## ПОКРОКОВЕ ВИКОНАННЯ ЗАВДАННЯ

Ми будемо створювати сайт-портфоліо - особисту веб-сторінку студента. Це зручний формат: він невеликий, але містить усі необхідні елементи (навігація, секції, форма зворотного зв'язку) і буде корисний вам у реальному житті.

### Крок 1. Підготовка структури проєкту

Перш ніж писати код, треба організувати файли. У веб-розробці прийнято зберігати всі файли одного проєкту в одній папці. Це важливо: якщо файли розкидані по різних папках - відносні шляхи (`href="style.css"`) не працюватимуть.

#### Завдання кроку: створити папку проєкту і файли в ній.

1. Натисніть правою кнопкою миші на Робочому столі → «Створити папку».
2. Назвіть її: `my-portfolio`  
УВАГА: лише латинські літери і дефіс! Пробіли, кирилиця, спецсимволи у назвах папок і файлів викликають помилки на серверах.
3. Відкрийте VS Code: меню File → Open Folder → оберіть `my-portfolio` → «Вибрати папку».
4. У лівій панелі (Explorer) ви бачите порожню папку `my-portfolio`.

Тепер створіть файли. У VS Code: натисніть іконку «New File» (аркушик зі плюсом) у верхній частині панелі Explorer або клацніть правою кнопкою → «New File»:

```
my-portfolio/
  index.html    ← ОБОВ'ЯЗКОВО така назва! (малими літерами)
  style.css     ← файл зі стилями
  script.js     ← файл з JavaScript (поки порожній)
  images/       ← папка для зображень (New Folder)
```

#### Чому ОБОВ'ЯЗКОВО `index.html`?

Коли ви відкриваєте сайт (наприклад, `github.io/my-portfolio/`), сервер шукає файл з назвою `index.html` як «головну сторінку». Це міжнародний стандарт, якого дотримуються всі веб-сервери. Якщо файл названо `home.html` або

main.html - сервер не знайде його і поверне помилку 404 Not Found.

## Крок 2. Базова HTML-структура

Відкрийте файл index.html (подвійний клік у панелі Explorer). Файл порожній. Є швидкий спосіб створити стандартну HTML-структуру: клацніть у порожньому файлі, введіть ! (знак оклику) і натисніть Tab або Enter. VS Code автоматично вставить шаблон.

Ви побачите такий код - давайте розберемо кожен рядок детально:

```
<!DOCTYPE html>
<!-- Рядок 1: повідомляє браузеру, що це HTML5, а не старіший HTML4 -->

<html lang="uk">
<!-- Рядок 2: початок документа. lang="uk" = мова - українська -->

<head>
<!-- HEAD - службова частина. Не відображається на сторінці. -->

    <meta charset="UTF-8">
    <!-- UTF-8 = кодування, що підтримує всі мови, зокрема українську -->

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <!-- Налаштування для мобільних: ширина = ширині пристрою, масштаб
1:1 -->

    <title>Портфоліо - Іван Іванов</title>
    <!-- Змініть на своє справжнє ім'я та прізвище! -->

    <link rel="stylesheet" href="style.css">
    <!-- Підключає наш файл style.css -->

</head>

<body>
<!-- BODY - весь видимий вміст сторінки -->

    <!-- Тут будемо додавати HTML-код -->

    <script src="script.js"></script>
    <!-- Script в кінці body: завантажується після всього HTML -->

</body>
</html>
```

Обов'язково виконайте: змініть текст у тегу <title> на своє справжнє ім'я та прізвище (наприклад: <title>Портфоліо - Марія Коваленко</title>). Збережіть файл: Ctrl+S.

## Крок 3. Створення навігаційної панелі

Навігаційна панель (navbar) - це горизонтальна смуга вгорі сайту з посиланнями. Вона дозволяє користувачу швидко перейти до будь-якої секції, не прокручуючи всю

сторінку. Ми зробимо її «липкою» (position: fixed) - вона завжди залишатиметься видимою вгорі, навіть коли прокручуєш сторінку.

Скопіюйте цей код і вставте у <body>, одразу після рядка <body> (але ДО рядка <script>):

```
<nav class="navbar">
  <div class="container">

    <!-- Логотип / назва сайту -->
    <div class="logo">Мое Портфоліо</div>

    <!-- Список посилань навігації -->
    <ul class="nav-links">
      <li><a href="#home">Головна</a></li>
      <li><a href="#about">Про мене</a></li>
      <li><a href="#skills">Навички</a></li>
      <li><a href="#projects">Проекти</a></li>
      <li><a href="#contact">Контакти</a></li>
    </ul>

    <!-- Кнопка-«гамбургер» для мобільних (три лінії) -->
    <div class="burger">
      <div class="line1"></div>
      <div class="line2"></div>
      <div class="line3"></div>
    </div>

  </div>
</nav>
```

Пояснення структури коду:

| Тег / клас              | Що це таке                                          | Навіщо потрібне                                                    |
|-------------------------|-----------------------------------------------------|--------------------------------------------------------------------|
| <nav class="navbar">    | Семантичний тег навігації з класом navbar           | Клас navbar дозволить застосувати CSS-стилі саме до цього елемента |
| <div class="container"> | Контейнер для обмеження ширини вмісту               | Щоб на широких моніторах вміст не розтягувався на весь екран       |
| <ul class="nav-links">  | Ненумерований список (ul = unordered list) посилань | Семантично правильно: навігація - це список пунктів                |
| href="#about"           | Посилання на якір (#) всередині сторінки            | При кліку браузер плавно прокрутить до елемента з id="about"       |
| <div class="burger">    | Кнопка-«гамбургер» (три горизонтальні лінії)        | Замінює меню на мобільних пристроях - класичний UI-патерн          |

## Крок 4. Підключення CSS: скидання стилів та змінні

Відкрийте файл style.css. Він порожній. Почнемо з двох важливих блоків, які є в кожному серйозному CSS-файлі: скидання стилів і CSS-змінні.

### Частина 4.1 - Скидання стилів (CSS Reset)

Кожен браузер (Chrome, Firefox, Safari) має власні «стилі за замовчуванням». Наприклад, Chrome додає зовнішні відступи до тегів <body>, <h1>, <p>. Firefox - трохи інші. Через це сайт може виглядати по-різному в різних браузерах. CSS Reset «обнуляє» всі ці відмінності:

```
/* ===== 1. СКИДАННЯ СТИЛІВ ===== */
/* * - «зоряний» селектор: означає «застосувати до ВСІХ елементів» */
* {
  margin: 0;           /* прибираємо зовнішні відступи */
  padding: 0;          /* прибираємо внутрішні відступи */
  box-sizing: border-box; /* розмір елемента включає padding і border */
}

/* box-sizing: border-box - дуже важлива властивість!
   Без неї: width: 200px + padding: 20px = фактична ширина 240px
   (неочікувано!)
   З нею: width: 200px + padding: 20px = фактична ширина 200px
   (логічно!) */
```

## Частина 4.2 - CSS-змінні

Після скидання оголошуємо CSS-змінні в блоці :root. Всі кольори, шрифти і розміри зберігаємо тут - щоб змінити дизайн, достатньо відредагувати цей блок:

```
/* ===== 2. CSS-ЗМІННІ (кольори та шрифти) ===== */
:root {
  /* Кольори (можете замінити на власну палітру) */
  --dark: #2c3e50; /* темно-синій: навібар, підвал */
  --accent: #3498db; /* синій акцент: посилання, кнопки */
  --light: #ecf0f1; /* світло-сірий: фон деяких секцій */
  --text: #333333; /* майже чорний: основний текст */
  --white: #ffffff;

  /* Шрифти */
  --font: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

  /* Розміри */
  --nav-height: 70px; /* висота навібару - будемо використовувати далі */
}

/* Базові стилі для всього сайту */
body {
  font-family: var(--font);
  color: var(--text);
  line-height: 1.6; /* відстань між рядками тексту - 160% від розміру шрифту */
}

/* Контейнер для обмеження ширини вмісту */
.container {
  max-width: 1200px; /* максимальна ширина - 1200px */
  margin: 0 auto; /* 0 зверху/знизу, auto зліва/справа = центрування */
  padding: 0 20px; /* 20px відступ зліва і справа (для маленьких екранів) */
}
```

## Крок 5. Стилiзацiя навігацiйної панелі

Тепер стилізуємо `<nav class="navbar">`. Додайте цей код у `style.css` після попереднього блоку:

```
/* ===== 3. НАВІГАЦІЙНА ПАНЕЛЬ ===== */
.navbar {
  background-color: var(--dark); /* темний фон */
  position: fixed; /* «прилипає» до верху: не рухається при прокрутці */
  top: 0; /* відстань від верху сторінки = 0 */
  left: 0; /* від лівого краю = 0 */
  width: 100%; /* на всю ширину вікна */
  height: var(--nav-height); /* висота = 70px (зі змінної) */
  z-index: 1000; /* «шар» відображення: 1000 = поверх усього іншого */
}

/* position: fixed пояснення:
   fixed = елемент виривається із звичайного потоку документа і
   позиціонується відносно вікна браузера (а не сторінки).
   z-index: якщо два елементи перекриваються, той з більшим z-index
   відображається поверх. */

.navbar .container {
  display: flex;
  justify-content: space-between; /* лого - ліворуч, меню -
  праворуч */
  align-items: center; /* по вертикалі - по центру */
  height: 100%; /* контейнер на всю висоту навібару */
}

.logo {
  color: var(--white);
  font-size: 1.5rem;
  font-weight: bold;
}

.nav-links {
  display: flex;
  list-style: none; /* прибираємо стандартні «кружечки» списку */
  gap: 30px; /* відстань між пунктами меню */
}

.nav-links a {
  color: var(--white);
  text-decoration: none; /* прибираємо підкреслення посилань */
  font-size: 1rem;
  transition: color 0.3s; /* плавна зміна кольору за 0.3 секунди */
}

.nav-links a:hover {
  color: var(--accent); /* при наведенні - синій колір */
}

/* Кнопка-«гамбургер» - поки приховуємо */
.burger {
  display: none;
  flex-direction: column;
  gap: 5px;
  cursor: pointer; /* курсор-рука при наведенні */
}
```

```
.burger div {
  width: 25px;
  height: 3px;
  background-color: var(--white);
}
```

Збережіть файл (Ctrl+S). Якщо Live Server запущений - браузер автоматично оновиться і ви побачите темну навігаційну панель у верхній частині сторінки.

## Крок 6. Медіа-запит для мобільних пристроїв

Останній обов'язковий блок - медіа-запит. Він повинен бути В КІНЦІ CSS-файлу, тому що CSS читається зверху вниз і правила, що йдуть пізніше, перекривають попередні при однаковій специфічності. Це і є «каскадність» у CSS.

```
/* ===== 4. АДАПТИВНІСТЬ: екрани шириною 768px і менше ===== */
@media (max-width: 768px) {

  /* Ховаємо горизонтальне меню */
  .nav-links {
    display: none;
  }

  /* Показуємо кнопку-«гамбургер» */
  .burger {
    display: flex;
  }

  /* Клас .active буде додаватися JavaScript-ом при кліку */
  .nav-links.active {
    display: flex;
    flex-direction: column; /* вертикальне меню */
    position: absolute;
    top: var(--nav-height); /* одразу під навібаром */
    left: 0;
    width: 100%;
    background-color: var(--dark);
    padding: 20px;
    gap: 15px;
  }

}
```

Збережіть файл. Тепер CSS-частина готова. Перейдемо до тестування.

## Крок 7. Локальне тестування у браузері

Перш ніж розгортати сайт у інтернеті, обов'язково перевірте його локально. «Локально» означає - на вашому комп'ютері, без інтернету.

### Запуск Live Server

Якщо Live Server ще не запущений: знайдіть файл index.html у панелі Explorer VS Code → клацніть правою кнопкою → «Open with Live Server». Або натисніть кнопку «Go Live» у правому нижньому куті рядка стану VS Code.

Браузер відкриється за адресою `http://127.0.0.1:5500` або `http://localhost:5500`. Це і є ваш локальний сервер.

### Що перевірити

Переконайтеся у наступному:

- Навігаційна панель видима вгорі сторінки і темного кольору
- Посилання меню відображаються горизонтально
- При наведенні на посилання - колір змінюється (це перевірка transition)
- Стили застосовано (не біла сторінка з чорним текстом - стандартний вигляд без CSS)

### Тестування мобільного вигляду

Щоб перевірити, як сайт виглядає на телефоні, не потрібен реальний телефон. Браузер має вбудований емулятор:

| Дія                       | Як виконати                                                                   | Що перевіряємо                                    |
|---------------------------|-------------------------------------------------------------------------------|---------------------------------------------------|
| Відкрити DevTools         | Натисніть F12 (або Ctrl+Shift+I)                                              | Відкривається панель інструментів розробника      |
| Увімкнути режим пристроїв | Клацніть іконку смартфона у верхньому лівому куті DevTools (або Ctrl+Shift+M) | Сторінка переходить у режим емуляції мобільного   |
| Обрати пристрій           | У верхньому випадаючому меню оберіть, наприклад, «iPhone 12»                  | Сторінка показується у розмірі вибраного пристрою |
| Перевірити гамбургер      | Ширина = 375px (iPhone) - кнопка-«гамбургер» повинна з'явитися замість меню   | Медіа-запит працює коректно                       |
| Перевірити великий екран  | Розширте вікно до >768px                                                      | Меню повертається, гамбургер зникає               |

## ПОРАДИ ТА ТИПОВІ ПОМИЛКИ

### Корисні поради

- Зберігайте файл (Ctrl+S) після кожного кроку - Live Server автоматично оновить браузер.
- Якщо щось пішло не так - натисніть Ctrl+Z для скасування останніх дій.
- Не обов'язково розуміти кожен рядок CSS одразу - головне, щоб сайт виглядав правильно. Розуміння прийде з практикою.

- Використовуйте коментарі (`/* ... */` у CSS і `<!-- ... -->` у HTML) - через тиждень ви скажете собі «дякую».
- Коли щось не працює - завжди спочатку перевірте консоль (DevTools → Console): там є повідомлення про помилки.

### Типові проблеми та їх рішення

| Проблема                                            | Можлива причина                                                                  | Рішення                                                                                                                                                                                                                                |
|-----------------------------------------------------|----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Стили не застосовуються (сторінка виглядає без CSS) | Помилка у рядку підключення CSS або неспівпадіння назви файлу                    | Перевірте: <code>&lt;link rel="stylesheet" href="style.css"&gt;</code> у <code>&lt;head&gt;</code> . Переконайтеся, що файл називається <code>style.css</code> (малими літерами!). Назви файлів чутливі до регістру на серверах Linux. |
| Live Server не запускається                         | Розширення не встановлено або не активоване                                      | Перевірте: <code>Ctrl+Shift+X</code> → введіть Live Server → «Install». Спробуйте: правою кнопкою на <code>index.html</code> → «Open with Live Server».                                                                                |
| Кнопка-«гамбургер» не зникає на великих екранах     | Використано <code>min-width</code> замість <code>max-width</code> у медіа-запиті | Перевірте медіа-запит: він повинен мати <code>@media (max-width: 768px)</code> <code>{ .burger { display: none; } }</code>                                                                                                             |
| Навігаційна панель перекриває вміст сторінки        | Не встановлено відступ зверху для <code>&lt;body&gt;</code>                      | Додайте у CSS: <code>body { padding-top: var(--nav-height); }</code>                                                                                                                                                                   |
| Українські літери відображаються як «кракозябри»    | Відсутній або неправильний charset                                               | Переконайтеся, що у <code>&lt;head&gt;</code> є: <code>&lt;meta charset="UTF-8"&gt;</code> . Збережіть файл у кодуванні UTF-8 (VS Code: правий нижній кут → «UTF-8»).                                                                  |

### ВИСНОВКИ

У ході виконання цієї лабораторної роботи студенти освоїли основи мови розмітки HTML5 і каскадних таблиць стилів CSS3, налаштували робоче середовище у редакторі VS Code з необхідними розширеннями, створили базову структуру проєкту для веб-сайту-портфоліо. Було вивчено семантичні теги HTML5, принципи Flexbox-розташування, механізм CSS-змінних та адаптивного дизайну з медіа-запитами. Набуті навички є фундаментом для подальшого розгортання сайту на хмарних платформах GitHub Pages та Netlify у наступних лабораторних роботах.

### 6. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке HTML і яка його роль у створенні веб-сторінок? Чим HTML відрізняється від мов програмування (наприклад, Python)?
2. Що таке CSS і як він пов'язаний з HTML? Наведіть приклад одного CSS-правила зі своєї роботи та поясніть кожну його частину.

3. Що таке CSS-змінні (Custom Properties) та яка їх практична перевага? Де у вашому проєкті вони використовуються?
4. Що таке адаптивний дизайн і навіщо він потрібен? Як реалізуються медіа-запити (@media)?
5. Чому файл з головною сторінкою обов'язково повинен називатися index.html? Що станеться, якщо назвати його home.html?
6. Що означає атрибут charset="UTF-8" у тегу <meta>? Що станеться, якщо його не вказати при роботі з українськими текстами?
7. Що таке position: fixed та де воно використовується у нашому проєкті? Як воно відрізняється від position: relative?
8. Що таке Flexbox? Наведіть приклад трьох CSS-властивостей з роботи, що стосуються Flexbox, та поясніть кожен.

## ПРАВИЛА ОФОРМЛЕННЯ ЗВІТУ

Звіт подається у вигляді документа Word (.docx) або PDF. Він повинен містити такі обов'язкові розділи:

| № | Розділ звіту                      | Що повинно бути                                                                                                                                                         | Бали |
|---|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 1 | Титульна сторінка                 | Назва університету, кафедра, назва і номер роботи, ПІБ студента, група, дата виконання                                                                                  | 5    |
| 2 | Мета роботи                       | Переформулюйте мету своїми словами (не копіюйте дослівно)                                                                                                               | 5    |
| 3 | Скріншоти виконання               | Не менше 5 скріншотів: структура папки у VS Code; код index.html; код style.css; вигляд сайту на великому екрані; вигляд сайту у режимі мобільного емулятора (DevTools) | 30   |
| 4 | Пояснення коду                    | До кожного скріншота - короткий опис: що зображено, що робить цей фрагмент коду                                                                                         | 20   |
| 5 | Відповіді на контрольні запитання | Власними словами, без копіювання з теоретичної частини                                                                                                                  | 30   |
| 6 | Висновки                          | 3–5 речень: що навчилися, які труднощі виникли, як вирішили                                                                                                             | 10   |

### Технічні вимоги до оформлення

- Шрифт: Times New Roman, розмір 14 пт для основного тексту, 16 пт - для заголовків розділів.
- Міжрядковий інтервал: 1.5.
- Поля: ліве - 30 мм, праве - 15 мм, верхнє і нижнє - 20 мм.
- Нумерація сторінок: внизу по центру, починаючи з другої сторінки.

- Скріншоти: вставляти як зображення, вирівняні по центру, з підписом «Рисунок X - Назва».
- Фрагменти коду у звіті оформляти моноширинним шрифтом (Courier New, 11 пт), виділяючи сірим фоном.
- Назви файлів, тегів, атрибутів і CSS-властивостей писати моноширинним шрифтом: `index.html`, `<nav>`, `display: flex`.

### **Що недопустимо**

- Копіювати код і пояснення з методички - відповіді повинні бути власними.
- Скріншоти чужих сайтів або зображення з інтернету замість власної роботи.
- Звіт без скріншотів виконання - не приймається.
- Файл з неправильним форматом (наприклад, .pages або Google Docs без конвертування).

# ЛАБОРАТОРНА РОБОТА № 3. ТЕМА: ПУБЛІКАЦІЯ ПЕРСОНАЛЬНОГО САЙТУ-ПОРТФОЛІО НА GITHUB PAGES

## Вступ до теми

GitHub - це не просто сховище коду. Це стандарт індустрії: практично кожна ІТ-компанія використовує Git та GitHub для командної розробки. GitHub Pages дає змогу без жодних витрат опублікувати готовий сайт, прямо з репозиторію. У цій роботі студент вперше знайомиться з Git - системою, яку використовують усі розробники світу.

Реальний кейс. Microsoft GitHub Pages активно використовується великими проєктами з відкритим кодом. Наприклад, офіційна документація Bootstrap (найпопулярнішої CSS-бібліотеки з понад 160 тисячами зірок на GitHub) розміщена саме на GitHub Pages. Компанія Airbnb публікує на GitHub Pages свої внутрішні дизайн-системи та стайлгайди для розробників.

## Мета роботи

Навчитися реєструватися на платформі GitHub та створювати репозиторії; опанувати базові команди системи контролю версій Git; розробити повноцінний персональний сайт-портфоліо з використанням HTML5, CSS3 та JavaScript; опублікувати сайт безкоштовно на GitHub Pages; навчитися оновлювати сайт після внесення змін.

## Завдання до роботи

Студент реєструється на GitHub, налаштовує репозиторій спеціального вигляду для Pages, завантажує файли сайту, активує хостинг та перевіряє доступність сайту за публічним посиланням.

## Теоретичні відомості

### Портфоліо веб-розробника

Портфоліо - це збірка кращих робіт, яка демонструє практичні вміння потенційним роботодавцям або клієнтам. У веб-розробці портфоліо - це особистий сайт, де показано реальні проєкти, навички та контактні дані.

## 2. ТЕОРЕТИЧНІ ВІДОМОСТІ

### Чому ця тема є у вашій дисципліні

GitHub - це не просто сховище коду. Це **стандарт індустрії**: практично кожна ІТ-компанія використовує Git і GitHub для командної розробки. Тому знання Git та вміння

публікувати проекти - це перша практична навичка, яка буде потрібна вам вже на стажуванні або першій роботі.

GitHub Pages дає змогу без жодних витрат опублікувати готовий сайт прямо з репозиторію. Тобто після виконання цієї лабораторної роботи у вас з'явиться **реальна адреса в інтернеті** - наприклад, <https://ivanpetrenko.github.io> - яку можна показати будь-кому.

### **Зверніть увагу:**

Найважливіше: у цій роботі ви вперше знайомитеся з Git - системою, яку використовують усі розробники світу.

Так, перші команди в терміналі можуть здаватися страшними - але інструкція покрокова, і після першого успішного push стає набагато легше.

### **Портфоліо веб-розробника - навіщо воно потрібне**

Портфоліо - це збірка кращих робіт, яка демонструє ваші практичні вміння потенційним роботодавцям або клієнтам. У веб-розробці портфоліо - це **особистий сайт**, де показані реальні проекти, навички та контактні дані. Для студента першого курсу портфоліо - це стартова точка: навіть один студентський проєкт вже говорить більше, ніж слова «я знаю HTML».

| <b>Без портфоліо</b>                                  | <b>З портфоліо</b>                                            |
|-------------------------------------------------------|---------------------------------------------------------------|
| «Я знаю HTML та CSS» - просто слова без підтвердження | Живий сайт - підтвердження того, що ви вмієте писати код      |
| Роботодавець не може перевірити рівень до співбесіди  | Видно стиль коду, уважність до деталей, дизайн                |
| На 30–40% менше відгуків на вакансії                  | На 30–40% більше відгуків (за даними досліджень ринку праці)  |
| Немає чого показати на співбесіді - незручна пауза    | «Ось мій сайт, ось мої проєкти» - конкретна, впевнена розмова |

*Таблиця 2.1 - Переваги наявності онлайн-портфоліо*

### **Git - система контролю версій**

Git - це **розподілена система контролю версій**, яка відстежує зміни у файлах і дозволяє повертатися до будь-якої попередньої версії проєкту. Уявіть, що кожного разу, коли ви натискаєте Ctrl+S, Git робить «знімок» всіх ваших файлів. Якщо щось зламалось - можна повернутись до будь-якого попереднього «знімку».

GitHub - це **хмарна платформа** для зберігання коду з використанням Git. Якщо Git - це ваш особистий щоденник змін, то GitHub - це сейф у хмарі, де щоденник зберігається і може бути доступний з будь-якого комп'ютера або для вашої команди.

### Основні поняття Git - простою мовою

| Термін     | Що означає                                                       | Аналогія з реальним світом                                        |
|------------|------------------------------------------------------------------|-------------------------------------------------------------------|
| repository | Сховище коду з усією історією змін                               | Папка з проєктом + журнал всіх версій                             |
| commit     | Зафіксований «знімок» проєкту в певний момент часу з описом змін | Збереження гри: будь-коли можна повернутись до цієї точки         |
| branch     | Незалежна копія коду. Основна гілка - main                       | Паралельний всесвіт: можна змінювати код, не торкаючись основного |
| push       | Відправлення локальних змін з комп'ютера на GitHub               | Завантажити файл у хмарне сховище (як Google Drive, але для коду) |
| clone      | Скопіювати репозиторій з GitHub на свій комп'ютер                | Завантажити чужий проєкт собі для вивчення або роботи             |

Таблиця 2.2 - Основні поняття Git

### Основні команди Git - детальне пояснення

Команди вводяться у термінал (командний рядок). Кожна команда починається зі слова git - це вказує системі, що ми звертаємось до Git, а не до іншої програми:

```
git init           # ініціалізація: «зробити цю папку репозиторієм»
git add .          # додати ВСІ файли до наступного коміту
git add index.html # додати лише один файл
git commit -m "опис змін" # зафіксувати зміни з обов'язковим описом
git remote add origin URL # прив'язати локальний репо до GitHub
git push -u origin main # відправити код на GitHub (перший раз)
git push           # відправити код на GitHub (всі наступні рази)
git status         # перевірити: які файли змінені, не додані тощо
git log --oneline  # переглянути список всіх комітів
```

Пояснення:

Що означає крапка у команді git add . ?

Крапка (.) означає «поточна папка і всі вкладені». Тобто `git add .` додає до наступного коміту всі файли і папки у вашому проєкті. Якщо хочете додати лише один файл - вкажіть його назву: `git add index.html`

### GitHub Pages - безкоштовний хостинг для студентів

GitHub Pages - це безкоштовний сервіс хостингу **статичних сайтів** від компанії GitHub. **Статичний сайт** - це сайт, де сервер роздає готові файли HTML, CSS і JavaScript без динамічної генерації на сервері. Наш сайт-портфоліо саме такий - він складається з html, css та js файлів.

| Характеристика     | Деталі                                     | Що це означає для вас                              |
|--------------------|--------------------------------------------|----------------------------------------------------|
| Вартість           | Повністю безкоштовно, без кредитної картки | Платити нічого не потрібно                         |
| URL сайту          | username.github.io                         | Замість username - ваш логін GitHub                |
| HTTPS              | Автоматично вмикається                     | Браузер не показує попередження «небезпечний сайт» |
| Ліміт трафіку      | 10 ГБ на місяць                            | Для портфоліо більш ніж достатньо                  |
| Розмір репозиторію | До 1 ГБ                                    | Наш проєкт займає < 1 МБ                           |
| Підтримка          | Лише статичні файли: HTML, CSS, JS         | Python/PHP/Node.js сервер НЕ підтримується         |
| Оновлення          | 1–5 хвилин після <code>git push</code>     | Зробили push - через кілька хвилин сайт оновиться  |

Таблиця 2.3 - Характеристики GitHub Pages

### Що таке термінал і як з ним працювати

Термінал (командний рядок) - це **текстовий інтерфейс** для керування комп'ютером. Замість кліків мишею - текстові команди. Для веб-розробника вміння працювати з терміналом є обов'язковим навиком.

| Операційна система | Як відкрити термінал                                                                         | Що побачите                                                        |
|--------------------|----------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| Windows 10/11      | Натисніть Win+R → введіть <code>cmd</code> → Enter.<br>Або: клацніть правою кнопкою на папці | Чорне або біле вікно з текстом типу <code>C:\Users\Ім'я&gt;</code> |

| Операційна система    | Як відкрити термінал                               | Що побачите                                  |
|-----------------------|----------------------------------------------------|----------------------------------------------|
|                       | проєкту → «Відкрити термінал» / «Open in Terminal» |                                              |
| macOS                 | Cmd+Пробіл → введіть «Terminal» → Enter            | Вікно з рядком типу ім'я@комп'ютер %         |
| VS Code (будь-яка ОС) | Меню Terminal → New Terminal (або Ctrl+` )         | Термінал відкривається прямо у VS Code внизу |

Таблиця 2.4 - Як відкрити термінал

**Рекомендація для початківців:** використовуйте термінал у VS Code (Ctrl+`). Він автоматично відкривається у вашій папці проєкту - не треба переходити до неї командою `cd`. Це зручно і зменшує кількість помилок.

#### **Корисна команда:**

Якщо ви хочете перейти до папки проєкту вручну, використовуйте команду:

`cd C:\Users\Ім'я\Desktop\my-portfolio` (Windows)

`cd ~/Desktop/my-portfolio` (macOS / Linux)

`cd` означає «change directory» - змінити поточну папку.

## **ПОКРОКОВЕ ВИКОНАННЯ ЗАВДАННЯ**

У цьому розділі ми крок за кроком пройдемо від реєстрації на GitHub до живого сайту в інтернеті. Кожен крок пронумерований і містить точні інструкції. **Не пропускайте кроки** - кожен наступний залежить від попереднього.

### **Крок 1. Реєстрація на GitHub**

#### **Мета кроку:**

Завдання кроку: створити обліковий запис на GitHub.com з правильним username.

#### **Перед реєстрацією - важливе правило щодо вибору username:**

Username стане частиною адреси вашого сайту: `username.github.io`. Тому обирайте відповідально:

- Використовуйте своє справжнє ім'я або прізвище: `ivanpetrenko`, `mkowalenko`
- Уникайте: `gamer123`, `superman777`, `coolboy2005` - виглядає непрофесійно у резюме
- Лише латинські літери, цифри та дефіс. Пробіли неможливі

- Username не можна легко змінити - якщо змінете, адреса сайту зміниться

### Покрокова реєстрація:

1. Відкрийте браузер і перейдіть на сайт: <https://github.com>
2. Натисніть велику зелену кнопку «Sign up» (угорі праворуч)
3. Введіть свою електронну пошту і натисніть «Continue»
4. Придумайте пароль (мінімум 8 символів, включно з цифрою та буквою) → «Continue»
5. Введіть бажаний username (перевіряйте доступність - система одразу покаже, зайнятий він чи ні) → «Continue»
6. Пройдіть перевірку «Я не робот» (гра-пазл) → «Create account»
7. На вашу пошту прийде лист з кодом підтвердження - введіть його
8. Оберіть план: «Free» (безкоштовний) → підтвердіть

## Крок 2. Встановлення Git на комп'ютер

### Мета кроку:

Завдання кроку: встановити програму Git, щоб можна було використовувати команди git у терміналі.

Git - це програма, яка встановлюється на ваш комп'ютер. **Без неї команди git init, git commit тощо не будуть працювати.**

| Операційна система | Як перевірити, чи Git встановлений                                                                                  | Де скачати Git                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| Windows            | Введіть у терміналі: <code>git --version</code> . Якщо бачите щось типу <code>git version 2.x</code> - встановлений | <a href="https://git-scm.com">https://git-scm.com</a> (кнопка Download for Windows)                                  |
| macOS              | Введіть у Terminal: <code>git --version</code> . На macOS Git зазвичай вже встановлений                             | Якщо немає - <a href="https://git-scm.com">https://git-scm.com</a> або через Homebrew: <code>brew install git</code> |

Таблиця 3.1 - Перевірка та встановлення Git

**Після встановлення Git (лише один раз!) налаштуйте своє ім'я та пошту - Git буде підписувати ними ваші коміти:**

```
git config --global user.name "Іван Петренко"
```

```
git config --global user.email "ivan@example.com"
```

```
# Перевірити налаштування:
```

```
git config --list
```

### **Важливо:**

Ці команди виконуються ОДИН РАЗ для всього комп'ютера (не для кожного проєкту).

Ваше ім'я та пошта будуть відображатися в кожному коміті. Вводьте реальні дані.

## **Крок 3. Створення репозиторію GitHub Pages**

### **Мета кроку:**

Завдання кроку: створити репозиторій на GitHub з правильною назвою для GitHub Pages.

**Найважливіше правило цього кроку:** для GitHub Pages репозиторій **ОБОВ'ЯЗКОВО** повинен називатися: `username.github.io` - де `username` це точно той `username`, що ви вказали при реєстрації (лише малі літери, без пробілів).

### **УВАГА - критично важливо:**

Якщо ваш `username` - `ivanpetrenko`, то репозиторій повинен називатись: `ivanpetrenko.github.io`

Якщо назва неправильна (наприклад, `ivan.github.io` або `ivanpetrenko.github.com`) - GitHub Pages НЕ активується автоматично.

Назва чутлива до регістру: `IvanPetrenko.github.io`  $\neq$  `ivanpetrenko.github.io`

### **Покрокове створення репозиторію:**

1. Увійдіть у свій акаунт на `github.com`
2. У правому верхньому куті натисніть «+» → «New repository» (або на головній сторінці - велика зелена кнопка «New»)
3. Description (опис) - за бажанням, наприклад: «Мое особисте портфоліо»
4. Visibility: оберіть «Public» (публічний). Без цього GitHub Pages не працюватиме
5. Поставте галочку «Add a README file» - це створить файл `README.md`
6. Натисніть «Create repository»

## **Крок 4. Підготовка файлів проєкту**

### **Мета кроку:**

Завдання кроку: переконатись, що файли з Лабораторної роботи № 2 готові до завантаження.

Якщо ви виконали Лабораторну роботу № 2, у вас вже є папка my-portfolio з файлами. Перед завантаженням переконайтеся, що структура папки правильна:

my-portfolio/

├── index.html ← **ОБОВ'ЯЗКОВО!** Головна сторінка

├── style.css ← Файл стилів

├── script.js ← JavaScript-файл

└── images/ ← Папка з зображеннями (якщо є)

| Що перевірити          | Правильно                                 | Неправильно (викличе проблеми)                                 |
|------------------------|-------------------------------------------|----------------------------------------------------------------|
| Назва головного файлу  | index.html                                | home.html, main.html, Index.html                               |
| Кодування файлів       | UTF-8 (VS Code показує внизу праворуч)    | Windows-1251 або ANSI - кирилиця «ламається»                   |
| Назви файлів і папок   | Лише латиниця, цифри, дефіс, підкреслення | Пробіли, кирилиця, спецсимволи - викликають помилки на сервері |
| Підключення CSS у HTML | <link rel="stylesheet" href="style.css">  | Відсутнє підключення або неправильний шлях                     |

Таблиця 3.2 - Чек-ліст перевірки файлів перед завантаженням

## Крок 5. Завантаження файлів на GitHub

### Мета кроку:

Завдання кроку: завантажити файли сайту на GitHub двома можливими способами.

**Існує два способи завантажити файли. Оберіть той, що зручніший:**

### **Спосіб А - Через веб-інтерфейс GitHub (для початківців)**

*Цей спосіб не вимагає знання команд Git. Підходить для першого разу:*

1. Відкрийте сторінку свого репозиторію на github.com
2. Натисніть кнопку «Add file» (кнопка з плюсом, справа) → «Upload files»
3. Відкрийте папку my-portfolio на своєму комп'ютері
4. Перетягніть ВСІ файли (index.html, style.css, script.js) і папку images прямо у вікно браузера (у зону «Drag files here»)
5. Прокрутіть вниз до блоку «Commit changes»
- 6.

7. Переконайтесь, що вибрано «Commit directly to the main branch»
8. Натисніть «Commit changes»

### **Спосіб Б - Через термінал Git (рекомендований спосіб)**

*Цей спосіб швидший для подальших оновлень. Виконайте по черзі ці команди у терміналі:*

# 1. Відкрийте термінал у папці my-portfolio (або перейдіть туди):

```
cd C:\Users\Ім'я\Desktop\my-portfolio # Windows
```

```
cd ~/Desktop/my-portfolio # macOS/Linux
```

# 2. Ініціалізуйте Git-репозиторій у цій папці:

```
git init
```

# 3. Додайте всі файли до наступного коміту:

```
git add .
```

# 4. Зробіть перший коміт з описом:

```
git commit -m "Initial commit: add portfolio website files"
```

# 5. Прив'яжіть до вашого GitHub-репозиторію (замініть USERNAME!):

```
git remote add origin https://github.com/USERNAME/USERNAME.github.io.git
```

# 6. Перейменуйте гілку на main (сучасний стандарт):

```
git branch -M main
```

# 7. Відправте код на GitHub:

```
git push -u origin main
```

### **Пояснення команди:**

Що означає -u у команді `git push -u origin main`?

Ключ -u (скорочення від --set-upstream) означає: «запам'ятай, що гілка main пов'язана з origin/main на GitHub».

Після першого push з -u всі наступні рази можна просто писати `git push` - без додаткових параметрів.

### **Крок 6. Активація GitHub Pages**

#### **Мета кроку:**

Завдання кроку: увімкнути публікацію сайту з репозиторію.

#### **Покрокова активація:**

1. Перейдіть на сторінку вашого репозиторію на [github.com](https://github.com)
2. Натисніть вкладку «Settings» (налаштування) - остання у верхньому рядку вкладок
3. У лівій бічній панелі знайдіть розділ «Code and automation» → натисніть «Pages»

4. У блоці «Build and deployment» → «Source» оберіть «Deploy from a branch»
5. У полі «Branch» оберіть «main», у другому випадяючому меню - «/ (root)»
6. Натисніть кнопку «Save»
7. Зачекайте 1–5 хвилин і оновіть сторінку налаштувань (F5)

| Стан          | Що відображається на вкладці<br>Settings → Pages | Що робити                                          |
|---------------|--------------------------------------------------|----------------------------------------------------|
| Готується...  | Жовта плашка: «GitHub Pages source saved»        | Зачекати 1–5 хвилин і оновити                      |
| Опубліковано! | Зелена / блакитна плашка з посиланням            | Перейти за посиланням - сайт готовий!              |
| Помилка       | Червона плашка або нічого немає                  | Перевірте назву репозиторію і наявність index.html |

Таблиця 3.3 - Стани публікації GitHub Pages

#### Крок 7. Перевірка живого сайту

##### Мета кроку:

Завдання кроку: переконатися, що сайт відкривається коректно у браузері.

**Що перевірити після відкриття сайту за адресою <https://username.github.io>:**

| Що перевіряємо      | Очікуваний результат                         | Якщо щось не так                                       |
|---------------------|----------------------------------------------|--------------------------------------------------------|
| Навігаційна панель  | Темна панель вгорі з пунктами меню           | Перевірте підключення style.css у HTML                 |
| Стилі CSS           | Сторінка виглядає з оформленням, не як текст | Перевірте, чи файл style.css завантажено у репозиторій |
| Зображення          | Фотографії завантажуються і відображаються   | Перевірте регістр назви файлу: photo.jpg ≠ Photo.jpg   |
| Мобільна версія     | Відкрийте на телефоні - сайт адаптується     | Перевірте медіа-запити у CSS                           |
| Посилання навігації | При кліку - плавна прокрутка до секцій       | Перевірте збіг href="#id" та id="id" у HTML            |

Таблиця 3.4 - Чек-ліст перевірки живого сайту

## Крок 8. Оновлення сайту після змін

### Мета кроку:

Завдання кроку: навчитися вносити зміни і оновлювати сайт у три команди.

Коли ви відредагуєте код і захочете побачити зміни на живому сайті - потрібно **синхронізувати** локальні файли з GitHub. Це робиться трьома командами (виконувати у терміналі, у папці проєкту):

# 1. Додати всі змінені файли:

```
git add .
```

# 2. Зробити коміт з описом змін:

```
git commit -m "Update: виправив мобільне меню"
```

# 3. Відправити на GitHub:

```
git push
```

# Через 1–3 хвилини зміни відобразяться на <https://username.github.io>

### Правила написання описів комітів (хороші звички з першого дня):

| Поганий опис коміту | Хороший опис коміту                         | Чому важливо                              |
|---------------------|---------------------------------------------|-------------------------------------------|
| update              | feat: додав секцію «Проекти»                | Зрозуміло, що саме зроблено               |
| fix                 | fix: виправив помилку у медіа-запиті @768px | Видно, що і де виправлено                 |
| changes             | style: змінив кольорову схему на темну      | Через місяць легко знайти потрібну версію |
| aaa / test / qqq    | docs: оновив README з описом проєкту        | Колеги розуміють без пояснень             |

Таблиця 3.5 - Приклади описів комітів

## 4. ПОРАДИ ТА ТИПОВІ ПОМИЛКИ

### Корисні поради

- Пишіть `git status` перед кожним `commit` - він покаже, які файли змінені, а які ні. Це допоможе уникнути «випадкових» комітів.
- Якщо термінал видає помилку - уважно прочитайте її. GitHub пише дуже конкретно: «repository not found», «permission denied» - кожна помилка має підказку.
- Зберігайте URL свого сайту у закладках і на початку кожного заняття перевіряйте, що він відкривається.
- Перш ніж `git push` - переконайтеся, що сайт коректно відображається локально (через Live Server у VS Code).
- Якщо сайт не з'явився через 10 хвилин після `push` - перевірте Settings → Pages: чи активоване розгортання з гілки main.

### Типові проблеми та їх вирішення

| Проблема                                             | Можлива причина                                                           | Рішення                                                                                                                                                  |
|------------------------------------------------------|---------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Сайт показує 404 Not Found</b>                    | Репозиторій названо неправильно або файл не index.html                    | 1. Перевірте назву репозиторію:<br>username.github.io<br>2. Переконайтесь, що головний файл - index.html<br>3. Зачекайте 5–10 хвилин і оновіть Ctrl+F5   |
| <b>Сайт є, але без стилів (тільки текст)</b>         | Файл style.css не завантажено або неправильний шлях у <link>              | 1. Перевірте, чи є style.css у репозиторії<br><link rel="stylesheet" href="style.css"><br>2. Перістр: Style.css ≠ style.css                              |
| <b>git push просить логін/пароль або помилку 403</b> | GitHub прибрав підтримку паролів - потрібен Personal Access Token або SSH | GitHub: Settings → Developer settings → Personal access tokens → Generate new token<br>Або налаштуйте SSH-ключ.<br>Зверніться до викладача за допомогою. |
| <b>Зображення не відображаються на сайті</b>         | Помилка реєстру у назві файлу або папки                                   | Сервер Linux чутливий до реєстру!<br>photo.jpg ≠ Photo.jpg ≠ PHOTO.JPG<br>Перевірте:<br>src="images/photo.jpg" - реєстр має збігатись                    |
| <b>Зміни не з'являються після push</b>               | Браузер показує кешовану стару версію                                     | Натисніть Ctrl+Shift+Delete → очистити кеш.<br>Або відкрийте у режимі інкогніто: Ctrl+Shift+N<br>Зачекайте 2–3 хвилини після push.                       |
| <b>Помилка: «repository not found»</b>               | Неправильна URL у команді git remote add origin                           | Відкрийте репозиторій на github.com, натисніть «Code» → скопіюйте HTTPS посилання<br>Виконайте: git remote set-url origin <нова URL>                     |

## ВИСНОВКИ

Виконавши цю лабораторну роботу, студенти опанували **повний цикл розробки та публікації веб-сайту**: від підготовки файлів у VS Code до розгортання на хмарній платформі GitHub Pages. Було засвоєно базові команди системи контролю версій Git (`git init`, `add`, `commit`, `push`), що є обов'язковим інструментом сучасного розробника.

Навичка публікації сайту на GitHub Pages є першим кроком до реального досвіду роботи з хмарними платформами та командного розгортання проєктів. Набуті знання безпосередньо застосовуватимуться у всіх наступних лабораторних роботах та майбутній практиці.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке портфоліо та навіщо воно потрібне веб-розробнику? Чому важливо мати живий сайт, а не просто розповідати про свої навички?
2. Що таке Git? Поясніть різницю між Git та GitHub простими словами.
3. Що таке `commit` у Git? Чому важливо писати осмислені повідомлення до кожного коміту? Наведіть приклад хорошого і поганого опису коміту.
4. Яка послідовність трьох команд Git використовується для оновлення сайту після змін? Що робить кожна команда?
5. Чому репозиторій для GitHub Pages обов'язково повинен називатися `username.github.io`? Що станеться, якщо назвати його по-іншому?
6. Що таке GitHub Pages та які його основні характеристики? Які типи файлів він підтримує, а які ні?
7. Що означає команда `git add .` (з крапкою)? Чим вона відрізняється від `git add index.html`?
8. Яка різниця між першим `git push -u origin main` та наступними `git push`? Що означає ключ `-u`?

## 7. ПРАВИЛА ОФОРМЛЕННЯ ЗВІТУ

Звіт подається у вигляді документа Word (.docx) або PDF і повинен містити такі обов'язкові розділи:

| № | Розділ звіту                      | Що повинно бути                                                                                                                                                                 | Бали |
|---|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 1 | Титульна сторінка                 | Назва університету, кафедра, назва і номер роботи, ПІБ студента, група, дата виконання                                                                                          | 5    |
| 2 | Мета роботи                       | Переформулюйте мету своїми словами (не копіюйте дослівно)                                                                                                                       | 5    |
| 3 | Скріншоти виконання               | Не менше 6 скріншотів: профіль GitHub; репозиторій з файлами; налаштування Pages; сайт на великому екрані; сайт у DevTools (мобільний режим); вивід команди git log у терміналі | 30   |
| 4 | Пояснення скріншотів              | До кожного скріншота - короткий опис: що зображено, що відбувається на цьому кроці                                                                                              | 20   |
| 5 | Відповіді на контрольні запитання | Власними словами, без копіювання з теоретичної частини                                                                                                                          | 30   |
| 6 | Висновки                          | 3–5 речень: що навчились, які труднощі виникли, як вирішили                                                                                                                     | 10   |

Таблиця 7.1 - Структура та оцінювання звіту

### Обов'язкові посилання у звіті

- Посилання на GitHub-репозиторій:  
<https://github.com/username/username.github.io>
- Посилання на живий сайт: <https://username.github.io>

### Технічні вимоги до оформлення

- Шрифт: Times New Roman, розмір 14 пт для основного тексту, 16 пт - для заголовків розділів.
- Міжрядковий інтервал: 1.5.
- Поля: ліве - 30 мм, праве - 15 мм, верхнє і нижнє - 20 мм.
- Нумерація сторінок: внизу по центру, починаючи з другої сторінки.
- Скріншоти: вставляти як зображення, вирівняні по центру, з підписом «Рисунок X - Назва».

- Фрагменти коду у звіті оформляти моноширинним шрифтом (Courier New, 11 пт) з сірим фоном.

**Що недопустимо**

- Скріншоти чужих сайтів або зображення з інтернету замість власної роботи.
- Звіт без скріншотів виконання - не приймається.
- Відсутні посилання на репозиторій і живий сайт.

# ЛАБОРАТОРНА РОБОТА № 4. ТЕМА: РОЗГОРТАННЯ ВЕБ-САЙТУ НА ПЛАТФОРМІ NETLIFY. БАЗОВІ МОЖЛИВОСТІ

## Вступ до теми

Netlify - це наступний крок після GitHub Pages. Він автоматично оновлює сайт при кожній зміні коду: ви робите git push, а Netlify сам публікує нову версію. Це і є суть сучасного розгортання - не «завантажити файли вручну», а «налаштував один раз - і забув». Це підхід, який використовується у більшості сучасних команд веб-розробників.

Реальний кейс. Компанія Netlify була заснована у 2015 році двома розробниками, які хотіли вирішити проблему складного розгортання сайтів. Станом на 2024 рік платформа хостить понад 5 мільйонів сайтів, серед яких проєкти таких компаній як Peloton, Loblaw та Nike. Клієнти Netlify відзначають, що час від внесення змін до публікації скоротився з годин до хвилин - часто до 30-90 секунд.

## Мета роботи

Ознайомитися з хмарною платформою Netlify та її ключовими поняттями; навчитися публікувати статичний сайт двома способами (Drag & Drop та через GitHub); налаштувати автоматичне розгортання: зміна коду - автоматичне оновлення сайту; зрозуміти різницю між Netlify та GitHub Pages.

## Завдання до роботи

У ході виконання цієї роботи студент реєструється на Netlify, публікує сайт через Drag & Drop, налаштовує підключення до GitHub-репозиторію та перевіряє роботу автоматичного оновлення.

## Теоретичні відомості

### Що таке Netlify

Netlify (netlify.com) -хмарна платформа, що спеціалізується на хостингу статичних сайтів і реалізації JAMstack-архітектури. Компанію засновано у 2015 році; станом на 2024 рік платформа обслуговує понад 5 мільйонів сайтів.

Головна ідея Netlify -принцип «git push і забув». Розробник підключає GitHub-репозиторій, і далі платформа самостійно стежить за станом коду: щойно відправляються нові зміни, Netlify автоматично публікує оновлену версію сайту. Весь цикл від push до публікації займає 30–90 секунд. Це і є суть сучасного підходу до розгортання інформаційних

систем: не «завантажити файли вручну», а «налаштував конвеєр один раз -і він працює автономно».

### Основні поняття

Таблиця 1 -Основні поняття Netlify

| Поняття                       | Пояснення                                                                                                                |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| Deploy / Deployment (деплой)  | Перенесення сайту з комп'ютера розробника на хмарний сервер, щоб він став доступний усім користувачам мережі             |
| Автоматичне оновлення (CI/CD) | Зміна коду та git push ініціюють автоматичну перебудову і публікацію нової версії без ручних дій розробника              |
| Мережа доставки (CDN)         | Файли сайту зберігаються одночасно на 100+ серверах у різних країнах -сайт завантажується швидко з будь-якої точки світу |
| Build / Збірка                | Netlify перевіряє код і готує його до публікації; для чистого HTML/CSS це займає 10–30 секунд                            |
| Publish Directory             | Папка з файлами, які Netlify публікує. Для проєкту на чистому HTML -крапка «.» (корінь репозиторію)                      |
| Deploy Preview                | Тестова версія сайту зі змінами, яка не впливає на основний сайт; має окремий унікальний URL                             |
| Rollback (відкат)             | Повернення до будь-якої попередньої версії сайту одним кліком у Dashboard -якщо нова версія містить помилку              |
| netlify.toml                  | Необов'язковий файл конфігурації у корені репозиторію, що описує параметри збірки, переспрямування та заголовки          |

### Мережа доставки контенту (CDN)

Коли сайт публікується на Netlify, його файли автоматично реплікуються на 100+ серверів у різних країнах. Запит користувача направляється до географічно найближчого вузла, а не через весь океан до єдиного сервера. Це суттєво скорочує час відгуку та підвищує стійкість системи до відмов.

Таблиця 2 -Порівняння без CDN і з CDN

| Без CDN (один сервер) | З CDN (Netlify) |
|-----------------------|-----------------|
|-----------------------|-----------------|

|                                                     |                                                                        |
|-----------------------------------------------------|------------------------------------------------------------------------|
| Один сервер, наприклад у Каліфорнії (США)           | 100+ серверів: Франкфурт, Варшава, Токіо, Сінгапур...                  |
| Студент з Одеси -запит до США -затримка 500–1000 мс | Студент з Одеси -запит до Варшави -затримка 40–80 мс                   |
| Якщо сервер впав -сайт недоступний для всіх         | При недоступності одного вузла -запит автоматично переходить до іншого |

## Netlify та GitHub Pages: порівняння платформ

Таблиця 3 -Порівняння Netlify та GitHub Pages

| Характеристика                  | GitHub Pages                        | Netlify (безкоштовно)                            |
|---------------------------------|-------------------------------------|--------------------------------------------------|
| Призначення                     | Документація, прості особисті сайти | Сучасні фронтенд-застосунки, портфоліо, JAMstack |
| Глобальна мережа CDN            | Обмежена                            | Повноцінна, 100+ точок присутності               |
| Тестові версії (Deploy Preview) | Відсутні                            | Для кожного Pull Request автоматично             |
| Форми без бекенду               | Відсутні                            | 100 відправок на місяць безкоштовно              |
| Відкат до попередньої версії    | Через Git revert (ручний процес)    | Одним кліком у Dashboard (30 секунд)             |
| Ліміт трафіку                   | 10 ГБ/місяць                        | 100 ГБ/місяць                                    |
| Змінні середовища               | Відсутні                            | Повноцінна підтримка через Dashboard             |

## Netlify Forms -обробка форм без серверного коду

Netlify Forms -вбудована функція платформи, що дозволяє обробляти HTML-форми на статичному сайті без написання жодного рядка серверного коду (Python, Node.js тощо). Зазвичай форма зворотного зв'язку потребує власного серверу та бази даних. З Netlify Forms достатньо одного HTML-атрибута -і платформа бере на себе збирання, збереження та пересилання відправлених даних.

Безкоштовний план дозволяє обробляти 100 відправок на місяць. Дані зберігаються у Netlify Dashboard і можуть надходити на вказану email-адресу в режимі реального часу.

Обов'язкові умови для роботи Netlify Forms:

1. атрибут data-netlify="true" у тегу <form>;
2. атрибут name="назва" у тегу <form>;

3. приховане поле `<input type="hidden" name="form-name" value="назва">`, де value точно збігається з name у тегу `<form>`.

### Захист конфіденційних даних

Конфіденційні дані -паролі, API-ключі, токени доступу -категорично не можна записувати безпосередньо у файли JavaScript, які завантажуються на GitHub: вони будуть публічно видимі усім. Це порушення базових принципів безпеки інформаційних систем і може призвести до компрометації облікових записів.

Правильне рішення відповідно до стандартів безпеки: зберігати конфіденційні дані у Netlify Dashboard як змінні середовища (Environment Variables), а в коді використовувати лише посилання на них.

*Таблиця 4 -Правильне та неправильне зберігання конфіденційних даних*

| Неправильно (у коді)                                     | Правильно (через Netlify Dashboard)                                 |
|----------------------------------------------------------|---------------------------------------------------------------------|
| <code>const API_KEY = 'sk-abc123secret';</code>          | <code>const API_KEY = process.env.API_KEY;</code>                   |
| Ключ видимий усім у GitHub                               | Ключ зберігається лише у Netlify Dashboard, видимий тільки власнику |
| При компрометації -треба змінювати код і робити git push | При компрометації -змінюєте лише у Dashboard, код не чіпаєте        |

### Тестові версії (Deploy Previews)

Netlify може автоматично розгорнути тестову версію сайту для кожної нової гілки або Pull Request. Це дозволяє побачити реальний результат змін у браузері до того, як вони потраплять в основну версію сайту. Тестова версія має окремий унікальний URL і жодним чином не впливає на основний сайт. Це архітектурний патерн Blue-Green Deployment у спрощеному вигляді.

### Відкат до попередньої версії (Rollback)

Якщо нова версія сайту містить критичну помилку, Netlify дозволяє миттєво повернутися до будь-якої попередньої версії одним кліком у розділі Deploys. Процес займає 30 секунд і не потребує виправлення коду чи виконання команд у терміналі. Це забезпечує властивість відмовостійкості інформаційної системи.

### Netlify CLI -керування через термінал

Netlify CLI -інструмент командного рядка для управління сайтами Netlify безпосередньо з терміналу розробника. Встановлюється через npm (менеджер пакетів Node.js). CLI особливо корисний для тестування функцій Netlify (зокрема форм) локально, до публікації.

*Таблиця 5 -Основні команди Netlify CLI*

| Команда               | Що робить                                     | Коли використовувати                           |
|-----------------------|-----------------------------------------------|------------------------------------------------|
| netlify login         | Авторизація через браузер                     | Один раз після встановлення CLI                |
| netlify init          | Підключення папки проєкту до сайту на Netlify | На початку роботи з існуючим сайтом            |
| netlify dev           | Локальний сервер з усіма функціями Netlify    | Для тестування форм та змінних локально        |
| netlify deploy        | Деплой у тестову (Draft) версію               | Перед фінальним деплоєм - перевірити результат |
| netlify deploy --prod | Деплой в основний сайт                        | Коли все перевірено і готово до публікації     |
| netlify open:admin    | Відкрити Dashboard у браузері                 | Швидкий доступ до налаштувань                  |

## Конфігураційний файл netlify.toml

netlify.toml -необов'язковий, але важливий файл налаштувань у корені репозиторію. Він дозволяє описати всі параметри Netlify безпосередньо у коді: директорію публікації, версію Node.js, правила переспрямування (redirects) та HTTP-заголовки (headers). Перевага такого підходу полягає в тому, що налаштування зберігаються разом з кодом у системі контролю версій і не потрібно налаштовувати платформу вручну при кожному новому проєкті.

## ПОРЯДОК ВИКОНАННЯ РОБОТИ

### Крок 1. Підготовка та локальна перевірка проєкту

Для виконання цієї лабораторної роботи використовується сайт-портфоліо, розроблений у попередній лабораторній роботі. Переконайтеся, що папка my-portfolio містить файли index.html, style.css, script.js та папку images із зображеннями.

Перед розгортанням обов'язково виконайте локальну перевірку сайту:

1. відкрийте папку проєкту у VS Code;
2. запустіть розширення Live Server (правий клік на index.html → "Open with Live Server");
3. переконайтеся, що сайт відкривається без помилок у консолі браузера (F12);
4. перевірте, що всі зображення завантажуються коректно;
5. перевірте, що шляхи до файлів є відносними (style.css, а не C:\Users\...\style.css).

Локальна перевірка є обов'язковим кроком: помилки у коді, виявлені до розгортання, виправити значно простіше, ніж після публікації.

## Крок 2. Створення файлу конфігурації netlify.toml

У корені папки my-portfolio створіть новий файл з назвою netlify.toml. Цей файл не має розширення після крапки -лише ім'я. Відкрийте його у VS Code та введіть такий вміст:

```
# netlify.toml -файл конфігурації Netlify
[build]
publish = "."
[build.environment]
NODE_VERSION = "20"
[[redirects]]
from = "/*"
to = "/index.html"
status = 200
[[headers]]
for = "/*"
  [headers.values]
  X-Frame-Options = "DENY"
  X-Content-Type-Options = "nosniff"
```

Параметр publish = "." вказує, що публікується весь корінь репозиторію. Розділ [[redirects]] забезпечує коректну роботу навігації в односторінкових додатках. Розділ [[headers]] додає базові HTTP-заголовки безпеки -це хороша практика для будь-якого веб-проєкту.

Збережіть файл (Ctrl+S).

## Крок 3. Реєстрація на Netlify

Відкрийте браузер і перейдіть за адресою: <https://app.netlify.com/signup>. Виконайте наступні дії:

1. на сторінці реєстрації оберіть «Sign up with GitHub» -це пов'яже ваш обліковий запис Netlify з GitHub;
2. авторизуйтеся у GitHub, якщо потрібно;
3. після перенаправлення назад на Netlify дайте відповідь на питання про команду - оберіть «I'm just me»;
4. натисніть «Get started».

Реєстрація займає менше хвилини і не потребує введення платіжних даних. Безкоштовний план Netlify є достатнім для виконання всіх завдань цієї лабораторної роботи.

## Крок 4. Перший деплой методом Drag & Drop

Метод Drag & Drop дозволяє швидко опублікувати сайт без підключення до системи контролю версій. Він підходить для швидкої перевірки, але не зберігає зв'язок з кодом.

Виконайте такі дії:

1. у Netlify Dashboard прокрутіть сторінку вниз до блоку «Drag and drop your site output folder here»;
2. відкрийте Провідник Windows (Win+E) та знайдіть папку my-portfolio;
3. **ВАЖЛИВО:** перетягніть саму папку (my-portfolio), а не окремі файли з неї;
4. відпустіть папку у відведений області Dashboard.

Через 5–15 секунд Netlify опублікує сайт і надасть URL виду: random-name-123456.netlify.app. Перейдіть за цим посиланням і переконайтеся, що сайт відображається коректно. Зробіть знімок екрана для звіту.

### Крок 5. Зміна назви сайту

Netlify автоматично генерує випадкову назву для сайту. Щоб змінити її на зрозумілу:

1. у Dashboard натисніть «Site configuration»;
2. знайдіть розділ «Site name» та натисніть «Change site name»;
3. введіть власну назву, наприклад: mya-prizvysche або my-portfolio-2026;
4. правила для назви: лише малі латинські літери, цифри та дефіс (-); пробіли та кирилиця не допускаються;
5. натисніть «Save».

Якщо система повідомляє «already taken» -назва вже зайнята іншим користувачем. Додайте рік, ваш нікнейм або довільне число.

### Крок 6. Підключення GitHub та налаштування автоматичного оновлення

Автоматичне оновлення (CI/CD) означає: ви змінили код → зробили git push → Netlify автоматично оновлює сайт без будь-яких ручних дій. Це основна перевага Netlify перед ручним завантаженням файлів.

Виконайте підключення:

1. у Netlify Dashboard натисніть «Add new site» → «Import an existing project» → «GitHub»;
2. дозвольте Netlify доступ до ваших репозиторіїв (якщо запитує);
3. знайдіть репозиторій my-portfolio у списку та клацніть на ньому.

На сторінці налаштувань збірки встановіть такі параметри:

*Таблиця 6 -Параметри збірки для чистого HTML/CSS/JS*

| Поле              | Значення   | Пояснення                           |
|-------------------|------------|-------------------------------------|
| Branch to deploy  | main       | Гілка, за якою стежить Netlify      |
| Build command     | (порожнє)  | Для чистого HTML збірка не потрібна |
| Publish directory | . (крапка) | Публікується корінь репозиторію     |

Натисніть «Deploy site». Через 30–90 секунд статус зміниться на зелений «Published». Перейдіть за наданим URL та переконайтеся, що сайт доступний.

## Крок 7. Перевірка роботи автоматичного оновлення

Переконайтеся, що конвеєр CI/CD налаштований коректно. Для цього внесіть невелику зміну до коду:

1. відкрийте файл index.html у VS Code;
2. знайдіть тег <title> та змініть його вміст (наприклад, додайте «| v2»);
3. збережіть файл (Ctrl+S).

Відкрийте термінал у VS Code (Ctrl+`) та виконайте команди:

```
git add .
git commit -m "update: тест автоматичного оновлення"
git push
```

Поверніться у Netlify Dashboard -ви побачите новий деплой зі статусом «Building». Через 30–60 секунд сайт оновиться автоматично. Зробіть знімок екрана списку деплоїв для звіту.

## Крок 8. Налаштування Netlify Forms

Додайте обробку форми зворотного зв'язку без серверного коду. Відкрийте файл index.html та знайдіть тег <form> у секції контактів. Замініть його відповідно до прикладу нижче:

```
<!-- БУЛО: -->
<form class="contact-form" id="contactForm">

<!-- СТАЛО: -->
<form name="contact" method="POST" data-netlify="true"
      class="contact-form" id="contactForm">

  <!-- Обов'язкове приховане поле: -->
  <input type="hidden" name="form-name" value="contact">

  <!-- Решта полів залишається без змін -->
</form>
```

Збережіть зміни та відправте на GitHub:

```
git add .
git commit -m "feat: додав Netlify Forms для форми контактів"
git push
```

Зачекайте завершення деплою (30–60 секунд). Перейдіть на сайт, заповніть форму і натисніть кнопку відправки. Перевірте результат у Dashboard: ваш сайт → «Forms» (ліве меню) → форма «contact» → список отриманих повідомлень.

## Крок 9. Створення сторінки подяки

Додайте до форми атрибут `action`, що вказує сторінку, на яку переходить користувач після успішної відправки:

```
<form name="contact" method="POST" data-netlify="true"
      action="/thank-you" class="contact-form" id="contactForm">
```

Створіть новий файл `thank-you.html` у папці `my-portfolio`. Файл повинен містити стандартну HTML5-структуру з заголовком «Дякую за повідомлення!», коротким текстом підтвердження та кнопкою або посиланням для повернення на головну сторінку.

Збережіть та відправте зміни на GitHub. Після завершення деплою заповніть форму на сайті -ви маєте автоматично потрапити на сторінку подяки.

## Крок 10. Налаштування email-повідомлень про нові відправки форми

Налаштуйте автоматичне надсилання email при кожній новій відправці форми:

1. у Dashboard перейдіть: ваш сайт → Forms → назва форми (contact);
2. натисніть «Form notifications» → «Add notification» → «Email notification»;
3. введіть вашу email-адресу;
4. натисніть «Save».

Після цього кожна відправка форми на сайті генеруватиме автоматичний лист на вказану адресу.

## Крок 11. Захист конфіденційних даних через змінні середовища

Додайте тренувальні змінні середовища у Netlify Dashboard:

1. перейдіть: ваш сайт → «Site configuration» → «Environment variables» → «Add a variable»;
2. додайте першу змінну: Key = `SITE_TITLE`, Value = Ваше Ім'я та Прізвище -Портфоліо;
3. додайте другу змінну: Key = `CONTACT_EMAIL`, Value = `ваша.email@gmail.com`;
4. натисніть «Save».

Для локальної розробки змінні середовища зберігаються у файлі `.env` у корені проєкту. Цей файл ніколи не відправляється на GitHub -він містить конфіденційні дані. Додайте його до файлу `.gitignore`:

```
# Вміст файлу .gitignore
.env
.env.local
node_modules/
```

Перевірте, що файл `.env` доданий до `.gitignore` до першого `git push` із цим файлом.

## Крок 12. Тестова версія через нову гілку Git

Створіть тестову версію сайту зі змінами, що не впливатиме на основний сайт:

```
# Створити нову гілку
git checkout -b feature/color-test
```

```
# Відкрийте style.css і змініть основний колір:
```

```
# --primary-color: #6366f1; → --primary-color: #10b981;
```

```
git add .  
git commit -m "style: тест нового кольору"  
git push origin feature/color-test
```

Перейдіть на GitHub у ваш репозиторій. GitHub запропонує «Compare & pull request» -натисніть. Заповніть заголовок і опис змін, натисніть «Create pull request».

У Netlify Dashboard → Deploys ви побачите новий деплой з позначкою «Deploy Preview». Відкрийте URL тестової версії -ви побачите сайт з тестовим кольором. Основний сайт залишається незмінним. Зробіть знімок екрана URL тестової версії у браузері для звіту.

### Крок 13. Відкат до попередньої версії

Ознайомтеся з функцією відкату:

1. у Netlify Dashboard перейдіть: ваш сайт → «Deploys»;
2. знайдіть будь-який попередній деплой у списку;
3. натисніть на нього -відкриється сторінка деплою з детальною інформацією;
4. натисніть «Publish deploy» -Netlify миттєво переключить сайт на цю стару версію;
5. зробіть знімок екрана кнопки «Publish deploy» для звіту;
6. після ознайомлення з функцією поверніть актуальну версію тим самим способом.

### Крок 14. Встановлення та використання Netlify CLI

Встановіть Netlify CLI через термінал VS Code (Ctrl+`):

```
npm install -g netlify-cli  
netlify --version          # перевірка встановлення  
netlify login              # авторизація через браузер  
netlify init               # підключення папки до сайту  
netlify dev                # локальний сервер з підтримкою форм  
netlify deploy             # деплой у тестову версію  
netlify deploy --prod      # деплой в основний сайт
```

Зробіть знімок екрана терміналу з результатом виконання netlify --version та netlify status для звіту.

## ТИПОВІ ПРОБЛЕМИ ТА ЇХ ВИРІШЕННЯ

Таблиця 7 -Типові проблеми та способи їх вирішення

| Проблема                 | Причина                                                                      | Рішення                                                                           |
|--------------------------|------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| Сайт показує помилку 404 | Файл не називається index.html або Publish Directory налаштовано неправильно | Перевірте назву файлу (малими літерами) та значення Publish Directory у Dashboard |

|                                     |                                                                             |                                                                           |
|-------------------------------------|-----------------------------------------------------------------------------|---------------------------------------------------------------------------|
| Сайт відображається без стилів      | Файл style.css не завантажено на GitHub                                     | Перевірте репозиторій -чи є файл style.css                                |
| Зміни не з'являються після git push | Браузер показує кешовану версію                                             | Натисніть Ctrl+Shift+Delete або відкрийте у режимі інкогніто              |
| Форма не з'являється у Forms        | Відсутній data-netlify="true" або value у hidden input не збігається з name | Перевірте обидва атрибути одночасно                                       |
| Email-сповіщення не приходять       | Не налаштовано Form Notifications                                           | Forms → назва форми → Form notifications → Add notification → Email       |
| netlify login не завершується       | Блокується розширенням браузера                                             | Спробуйте інший браузер або відключіть uBlock/AdBlock                     |
| Файл .env потрапив на GitHub        | Файл не доданий до .gitignore до першого push                               | Вважайте ключі скомпрометованими, замініть їх; додайте .env до .gitignore |

## ВИМОГИ ДО ЗВІТУ

Звіт з лабораторної роботи повинен містити:

1. Знімок екрана Netlify Dashboard з опублікованим сайтом після Drag & Drop деплою.
2. Знімок екрана з URL виду <https://ваша-назва.netlify.app> у рядку браузера.
3. Знімок екрана розділу Settings з активованим підключенням GitHub.
4. Знімок екрана списку деплоїв -видно, що після git push запустився новий деплой.
5. Вміст файлу netlify.toml (скопювати текст або зробити знімок екрана).
6. Знімок екрана Netlify Dashboard → Forms → отримане тестове повідомлення форми.
7. Знімок екрана сторінки подяки thank-you.html після відправки форми.
8. Знімок екрана розділу Environment Variables у Netlify Dashboard.
9. Знімок екрана URL тестової версії (Deploy Preview) у браузері.
10. Знімок екрана списку деплоїв з кнопкою «Publish deploy».
11. Знімок екрана терміналу з результатом виконання netlify --version та netlify status.
12. Вміст файлу .gitignore.
13. Посилання на основний сайт на Netlify.
14. Відповіді на контрольні запитання.

## ВИСНОВКИ

У ході виконання цієї лабораторної роботи студенти отримали практичний досвід розгортання веб-сайту на хмарній платформі Netlify та ознайомилися як з базовими, так і з розширеними її можливостями.

На першому етапі було опановано два способи публікації статичного сайту: швидкий Drag & Drop для оперативної перевірки та підключення через GitHub-репозиторій з налаштуванням конвеєру CI/CD. Практично підтверджено, що після виконання git push сайт оновлюється автоматично протягом 30–60 секунд без жодних ручних дій. Вивчено переваги Netlify над GitHub Pages: повноцінна глобальна мережа доставки контенту (CDN), вищий ліміт трафіку (100 ГБ/місяць), тестові версії та зручний відкат версій.

На другому етапі студенти опанували розширені функції платформи: налаштування обробки HTML-форм без серверного коду через Netlify Forms, захист конфіденційних даних через змінні середовища (Environment Variables), практичне використання тестових версій (Deploy Previews) для безпечного тестування змін до їх публікації в основному сайті та функцію відкату (Rollback) для миттєвого відновлення попередньої версії. Також було встановлено та протестовано Netlify CLI -інструмент управління деплоями безпосередньо з терміналу.

Отримані навички безпосередньо відповідають реальним практикам сучасної веб-розробки та архітектурі сучасних інформаційних систем, що будуються на принципах JAMstack, CI/CD та сервіс-орієнтованого підходу.

# **ЛАБОРАТОРНА РОБОТА № 5. ТЕМА: ЗНАЙОМСТВО З ХМАРНИМИ ТЕХНОЛОГІЯМИ. РЕЄСТРАЦІЯ ТА БАЗОВА КОНФІГУРАЦІЯ AWS**

## **Вступ до теми**

Amazon Web Services (AWS) - найбільша у світі хмарна платформа, яку використовують такі компанії як Netflix, Airbnb, BMW та NASA. Розуміння AWS є ключовою навичкою для будь-якого сучасного IT-фахівця. Ця лабораторна робота є початком знайомства з реальним хмарним середовищем корпоративного рівня.

Реальний кейс. Компанія Netflix переніс всю свою інфраструктуру до AWS у 2016 році. До цього вони витрачали роки на підтримку власних дата-центрів. Після міграції Netflix може автоматично масштабуватися від 1 до 100 мільйонів одночасних переглядів під час великих подій (наприклад, фінал сезону популярного серіалу) без жодного ручного втручання інженерів.

## **Мета роботи**

Ознайомитись з поняттям хмарних обчислень; зареєструвати безкоштовний акаунт Amazon Web Services (AWS); навчитися орієнтуватися в консолі управління та налаштувати базову безпеку.

## **Завдання до роботи**

Студент реєструє акаунт AWS, знайомиться з основними сервісами консолі (S3, IAM), налаштовує двофакторну аутентифікацію та бюджетне сповіщення для захисту від непередбачених витрат.

## **Теоретичні відомості**

### **Що таке хмарні обчислення**

Уявіть, що замість купівлі власного генератора ви просто підключаєтеся до електромережі та платите лише за спожиту електроенергію. Хмарні обчислення - це те саме, але для комп'ютерних ресурсів: сервери, сховища та програми доступні через Інтернет за потребою, без потреби купувати власне обладнання.

AWS (Amazon Web Services) - найбільша хмарна платформа у світі. Для нових користувачів вона надає Free Tier (безкоштовний рівень) на 12 місяців. Free Tier включає: 750 годин EC2 мікро-інстансу щомісяця, 5 ГБ сховища S3, 25 ГБ DynamoDB та понад 60 інших безкоштовних сервісів.

## **Основні сервіси AWS**

| Сервіс     | Що робить                               | Аналог                              |
|------------|-----------------------------------------|-------------------------------------|
| EC2        | Віртуальні сервери (комп'ютери у хмарі) | Ваш особистий сервер                |
| S3         | Сховище файлів будь-якого розміру       | Google Drive для розробників        |
| IAM        | Управління доступом та безпекою         | Система паролів та дозволів         |
| Lambda     | Запуск коду без сервера                 | Функція, яка запускається за подією |
| CloudWatch | Моніторинг та логи                      | Журнал подій системи                |

## Детальне покрокове виконання - Варіант 1

### Крок 1. Відкриття сторінки реєстрації

Відкрийте браузер та перейдіть за адресою: <https://aws.amazon.com/free>. На сторінці ви побачите інформацію про безкоштовний рівень (Free Tier). Натисніть кнопку «Create a Free Account».

### Крок 2. Заповнення форми реєстрації

Введіть вашу особисту email-адресу (рекомендується не студентська, бо після навчання вона може бути деактивована). Придумайте надійний пароль (мінімум 8 символів, великі та малі літери, цифри). Введіть ім'я акаунту (наприклад: student-petrenko). Натисніть «Verify email address» та введіть код, який прийде на пошту.

### Крок 3. Тип акаунту та контактні дані

На сторінці «Contact Information» оберіть тип акаунту: Personal (Особистий). Заповніть контактні дані: ваше ім'я, прізвище, номер телефону, адреса. Натисніть «Continue».

### Крок 4. Введення платіжних даних

Введіть дані кредитної або дебетової картки. Це лише для верифікації особи: AWS знімає 1 долар, але одразу повертає. Реального списання не буде, якщо залишатися в межах Free Tier. Натисніть «Verify and Continue». Зробіть знімок екрана сторінки після успішного підтвердження картки.

### Крок 5. Верифікація телефону

Оберіть метод верифікації: «Text message (SMS)». Введіть ваш номер телефону у міжнародному форматі (для України: +380XXXXXXXXXX). Натисніть «Send SMS». Введіть код із SMS-повідомлення. Натисніть «Continue».

### **Крок 6. Вибір тарифного плану**

На сторінці «Select a support plan» оберіть: Basic support - Free. Це безкоштовний план без обмежень по часу. Натисніть «Complete sign up». Зробіть знімок екрана підтвердження успішної реєстрації.

### **Крок 7. Вхід до консолі та знайомство з регіоном**

Перейдіть за адресою: <https://console.aws.amazon.com>. Оберіть «Root user», введіть email та пароль. Після входу ви побачите головну сторінку AWS Management Console. Зробіть знімок екрана головної сторінки консолі. У правому верхньому куті оберіть регіон Europe (Frankfurt) - eu-central-1 - це найближчий до України сервер.

### **Крок 8. Налаштування безпеки - MFA**

У консолі AWS перейдіть до сервісу IAM. На головній сторінці IAM у розділі «Security recommendations» ви побачите попередження «Add MFA for root user». Натисніть «Add MFA» - «Assign MFA device». Введіть ім'я пристрою: «my-phone». Оберіть тип: «Authenticator app». Натисніть «Next». Встановіть на смартфоні Google Authenticator або Microsoft Authenticator, відскануйте QR-код та введіть два послідовні коди для підтвердження.

### **Крок 9. Створення IAM-користувача**

У сервісі IAM перейдіть до розділу «Users» - «Create user». Введіть ім'я користувача: student-user. Поставте галочку «Provide user access to the AWS Management Console». Оберіть «I want to create an IAM user». Введіть пароль. Натисніть «Next». На сторінці «Set permissions» оберіть «Attach policies directly» та додайте «AdministratorAccess». Натисніть «Create user». Збережіть дані для входу у файл.

### **Крок 10. Налаштування бюджетного сповіщення**

Увійдіть під кореневим акаунтом. Натисніть на ваше ім'я - «Billing and Cost Management» - «Budgets» - «Create budget». Оберіть шаблон «Zero spend budget» - він спрацює, щойно AWS починає щось нараховувати (навіть 0.01 долара). У полі «Email recipients» введіть вашу email-адресу. Натисніть «Create budget». Зробіть знімок екрана налаштованого бюджету зі статусом «Active».

### **Висновки до Варіанту 1**

Виконавши цю лабораторну роботу, студент ознайомився з найбільшою у світі хмарною платформою Amazon Web Services та її можливостями для розробників. Було зареєстровано безкоштовний акаунт AWS Free Tier, налаштовано двофакторну аутентифікацію для захисту акаунту, створено окремого IAM-користувача згідно з принципом мінімальних привілеїв та налаштовано бюджетне сповіщення для контролю витрат. Ці базові налаштування є основою для безпечної роботи з хмарними ресурсами в наступних лабораторних роботах.

## **Варіанти завдань**

Варіант 2. Зареєструвати AWS-акаунт та налаштувати IAM-користувача з обмеженими правами (лише S3 ReadOnly). Перевірити, що обмежений користувач не може видаляти файли з S3.

Варіант 3. Налаштувати AWS Cost Explorer та зробити скріншот дашборду з поточними витратами та прогнозом.

Варіант 4. Налаштувати CloudTrail для логування всіх дій в акаунті AWS та переглянути журнал подій за останній час.

Варіант 5. Ознайомитися з AWS Organizations та підготувати звіт про переваги організаційної структури для великих компаній.

Варіант 6. Налаштувати AWS Config для відстеження змін конфігурації ресурсів та перевірити, як система реагує на зміну налаштувань S3-бакету.

## **Корисні посилання та ресурси**

- AWS Free Tier - <https://aws.amazon.com/free>
- AWS Console - <https://console.aws.amazon.com>
- AWS Getting Started - <https://aws.amazon.com/getting-started/>
- AWS IAM User Guide -

<https://docs.aws.amazon.com/IAM/latest/UserGuide/>

# **ЛАБОРАТОРНА РОБОТА № 6. ТЕМА: ХМАРНЕ СХОВИЩЕ ТА СПІЛЬНА РОБОТА З ДОКУМЕНТАМИ**

## **Вступ до теми**

Хмарні сховища є невід'ємною частиною сучасної роботи в ІТ. Від організації командної документації до зберігання резервних копій - розуміння хмарних сховищ є базовою компетентністю. Студент практично освоює Google Drive та Microsoft OneDrive - найпоширеніші платформи у корпоративному середовищі.

Реальний кейс. Компанія Deloitte (одна з чотирьох найбільших консалтингових фірм світу) перейшла повністю на Google Workspace у 2017 році. Понад 200 тисяч співробітників у 150 країнах використовують Google Drive для спільної роботи. До переходу підготовка звіту займала 2-3 дні через пересилання файлів електронною поштою. Після переходу той самий процес скорочується до кількох годин завдяки одночасному редагуванню в реальному часі.

## **Мета роботи**

Навчитися ефективно використовувати хмарні сховища Google Drive та Microsoft OneDrive для зберігання файлів, організації спільної роботи та автоматичної синхронізації. Зрозуміти поняття «хмарне сховище», «спільний доступ», «синхронізація», «версіонування документів».

## **Завдання до роботи**

Студент організовує структуру папок у Google Drive, створює та форматує документ у Google Docs, налаштовує спільний доступ, ознайомлюється з версіонуванням та порівнює Google Drive з Microsoft OneDrive.

## **Теоретичні відомості**

### **Хмарне сховище - поняття та переваги**

Хмарне сховище - це сервіс, який дозволяє зберігати файли на серверах провайдера та отримувати до них доступ з будь-якого пристрою через Інтернет. На відміну від звичайної флешки або жорсткого диска, хмарне сховище завжди актуальне, доступне та захищене від втрати. Аналогія: звичайний жорсткий диск - як особистий сейф, доступний лише вдома. Хмарне сховище - як банківська комірка: доступна з будь-якої точки світу, захищена, з можливістю надати доступ іншим людям.

| Сервіс             | Безкоштовний<br>обсяг | Особливості                                     |
|--------------------|-----------------------|-------------------------------------------------|
| Google Drive       | 15 ГБ                 | Інтеграція з Google Docs, Sheets, Slides        |
| Microsoft OneDrive | 5 ГБ                  | Вбудований у Windows 10/11, інтеграція з Office |
| Dropbox            | 2 ГБ                  | Відмінна синхронізація, зручний десктоп-клієнт  |
| iCloud Drive       | 5 ГБ                  | Вбудований у macOS та iOS                       |

### Ключові можливості хмарних сховищ

Синхронізація - файли автоматично оновлюються на всіх пристроях. Відредагували документ на телефоні - зміни одразу з'являються на ноутбучі.

Спільний доступ - можна надати іншій людині доступ до файлу або папки. Рівні доступу: «Глядач» (тільки перегляд), «Коментатор» (може залишати коментарі), «Редактор» (повний доступ).

Версіонування - Google Drive зберігає всі попередні версії файлу. Можна переглянути, що було місяць тому, і відновити будь-яку версію.

### Детальне покрокове виконання - Варіант 1

#### Крок 1. Відкриття Google Drive та створення структури

Відкрийте браузер та перейдіть за адресою: <https://drive.google.com>. Увійдіть у свій обліковий запис Google. Натисніть кнопку «+ Новий» - «Папка» (Folder). Назвіть папку: Навчання\_2025. Відкрийте щойно створену папку подвійним кліком. В середині неї створіть ще три підпапки: Лабораторні\_роботи, Лекції, Особисті\_нотатки. Зробіть знімок екрана структури папок.

#### Крок 2. Завантаження файлів

Відкрийте папку Лабораторні\_роботи. Натисніть «+ Новий» - «Завантажити файл» (Upload file). Завантажте будь-який файл зі свого комп'ютера (наприклад, текстовий документ або зображення). Зробіть знімок екрана завантаженого файлу у папці.

#### Крок 3. Створення Google Docs документа

Відкрийте папку Лабораторні\_роботи. Натисніть «+ Новий» - «Google Документи» (Google Docs). Відкриється новий документ у браузері. Кладніть на «Без назви» вгорі та введіть назву: Звіт\_ЛР7\_[Ваше\_Прізвище]. Зверніть увагу: документ зберігається автоматично (видно напис «Збережено на Диску»). Напишіть у документі короткий звіт про виконання цієї лабораторної роботи (5-10 речень). Використайте форматування: зробіть

заголовок через меню «Формат» - «Стилі абзацу» - «Заголовок 1». Зробіть знімок екрана документа з вмістом.

#### **Крок 4. Налаштування спільного доступу**

Натисніть синю кнопку «Надати доступ» (Share) у правому верхньому куті. У вікні, що з'явилося, знайдіть розділ «Загальний доступ» (General access). Змініть значення з «Обмежений» на «Усі, хто має посилання» (Anyone with the link). Рівень доступу залиште «Переглядач» (Viewer). Натисніть «Копіювати посилання» (Copy link). Вставте посилання у звіт - викладач перевірить ваш документ за цим посиланням. Зробіть знімок екрана вікна налаштувань спільного доступу.

#### **Крок 5. Перегляд історії версій**

Внесіть будь-яку зміну в документ (наприклад, допишіть речення). Збережіть (Ctrl+S). Потім перейдіть: меню «Файл» - «Історія версій» - «Переглянути історію версій». У правій панелі побачите список версій з датою та часом збереження. Натисніть на будь-яку попередню версію - документ відкриється у стані на той момент. Зробіть знімок екрана панелі версій.

#### **Крок 6. Знайомство з Microsoft OneDrive**

Відкрийте браузер та перейдіть: <https://onedrive.live.com>. Увійдіть або зареєструйтеся через Microsoft-акаунт. Відкрийте Google Drive та OneDrive у двох вкладках браузера та заповніть у своєму звіті таблицю порівняння інтерфейсів за такими параметрами: безкоштовний обсяг, зручність створення документів, зручність надання спільного доступу, інтеграція з іншими сервісами.

#### **Висновки до Варіанту 1**

Виконавши цю лабораторну роботу, студент практично освоїв роботу з хмарними сховищами Google Drive та Microsoft OneDrive. Набуто навичок організації файлів у хмарному просторі, спільної роботи з документами, версіонування та налаштування рівнів доступу. Зроблено порівняльний аналіз двох провідних хмарних сховищ. Ці навички є обов'язковими у сучасній командній роботі в IT-індустрії.

#### **Варіанти завдань**

Варіант 2. Організувати спільну роботу над документом Google Docs між двома учасниками: один редагує, інший коментує. Зробити знімки екрана режиму спільного редагування у реальному часі.

Варіант 3. Налаштувати автоматичне завантаження файлів до Google Drive через десктопний клієнт. Продемонструвати синхронізацію між комп'ютером та браузером.

Варіант 4. Використати Google Sheets для ведення таблиці з даними про виконання лабораторних робіт. Побудувати діаграму та надати спільний доступ.

Варіант 5. Порівняти Dropbox та Google Drive: зареєструвати обидва акаунти, виконати аналогічні операції та написати детальний порівняльний звіт.

Варіант 6. Дослідити функцію офлайн-роботи Google Drive: налаштувати офлайн-режим, відредагувати документ без Інтернету та переконатися в синхронізації після підключення.

### **Корисні посилання та ресурси**

- Google Drive - <https://drive.google.com>
- Microsoft OneDrive - <https://onedrive.live.com>
- Google Workspace Learning Center - <https://support.google.com/a/users/>
- Google Docs Offline Chrome Extension -

<https://workspace.google.com/intl/en/products/docs/>

# **ЛАБОРАТОРНА РОБОТА № 7. ТЕМА: РОЗМІЩЕННЯ САЙТУ У ХМАРІ. СТАТИЧНИЙ ХОСТИНГ НА AMAZON S3**

## **Вступ до теми**

Amazon S3 (Simple Storage Service) є одним з найстаріших і найнадійніших сервісів AWS. Він використовується не лише для зберігання файлів, але й для хостингу статичних сайтів, дистрибуції медіа-контенту та резервного копіювання. Знання S3 є обов'язковим для будь-якого хмарного розробника.

Реальний кейс. Компанія Airbnb зберігає в Amazon S3 понад 10 мільярдів фотографій. При цьому їх система автоматично генерує до 20 різних версій кожного знімку (для різних розмірів екранів) та зберігає всі версії у S3. Використання S3 + CloudFront (CDN AWS) дозволяє Airbnb завантажувати зображення за 50-100 мілісекунд для користувачів з будь-якої точки світу.

## **Мета роботи**

Навчитися розміщувати готовий веб-сайт (HTML/CSS файли) у хмарному сховищі Amazon S3 та відкривати його з Інтернету. Зрозуміти поняття «хмарне сховище», «бакет», «публічний доступ».

## **Завдання до роботи**

Студент створює S3-бакет, налаштовує статичний хостинг та Bucket Policy для публічного доступу, завантажує файли сайту та перевіряє його доступність за публічним URL.

## **Теоретичні відомості**

### **Amazon S3 - основні концепції**

Amazon S3 (Simple Storage Service) - хмарне сховище для зберігання файлів. Уявіть S3 як Google Drive, але для розробників: ви завантажуєте файли, налаштовуєте доступ і отримуєте пряме посилання. Бакет (bucket) - це контейнер (папка) для файлів у S3. Кожен бакет має унікальну назву в межах всього AWS. Файли всередині бакету називаються об'єктами.

S3 підтримує режим «Static website hosting» - коли він увімкнений, бакет поводить себе як веб-сервер: при зверненні за URL повертає index.html.

## **Детальне покрокове виконання - Варіант 1**

### **Крок 1. Відкриття сервісу S3**

Увійдіть до AWS Console: <https://console.aws.amazon.com>. У рядку пошуку вгорі введіть «S3» та натисніть на результат «S3 - Scalable Storage in the Cloud». Переконайтеся, що у правому верхньому куті обрано регіон EU (Frankfurt).

## **Крок 2. Створення бакету**

Натисніть кнопку «Create bucket» (Створити бакет). У полі «Bucket name» введіть унікальну назву у форматі: my-site-[ваше-прізвище-малими-латинськими] (наприклад: my-site-petrenko). Назва повинна бути унікальною в межах усього AWS у всьому світі - якщо назва зайнята, додайте цифри (my-site-petrenko-2025). У розділі «Object Ownership» переконайтеся, що обрано «ACLs disabled». Залиште регіон EU (Frankfurt).

## **Крок 3. Вимкнення блокування публічного доступу**

Знайдіть розділ «Block Public Access settings for this bucket». ЗНИМІТЬ галочку з «Block all public access». З'явиться попередження - поставте галочку в полі підтвердження. Це необхідно для публічного хостингу сайту. Прокрутіть сторінку до кінця та натисніть «Create bucket». Зробіть знімок екрана щойно створеного бакету у списку.

## **Крок 4. Увімкнення статичного хостингу**

Натисніть на назву вашого бакету, щоб відкрити його. Перейдіть на вкладку «Properties» (Властивості). Прокрутіть сторінку донизу до розділу «Static website hosting». Натисніть «Edit». Оберіть «Enable» (Увімкнути). У полі «Index document» введіть: index.html. У полі «Error document» введіть: index.html. Натисніть «Save changes». Зробіть знімок екрана налаштувань з відображенням Bucket website endpoint (URL вашого майбутнього сайту).

## **Крок 5. Додавання Bucket Policy**

Перейдіть на вкладку «Permissions» (Дозволи). Знайдіть розділ «Bucket policy» та натисніть «Edit». У текстовий редактор вставте наступний JSON-документ, **ОБОВ'ЯЗКОВО** замінивши НАЗВА-БАКЕТУ на вашу реальну назву бакету:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "PublicRead",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::НАЗВА-БАКЕТУ/*"
  }]
}
```

Натисніть «Save changes». Зробіть знімок екрана збереженої політики.

## **Крок 6. Завантаження файлів сайту**

Поверніться до вашого бакету в AWS Console. Перейдіть на вкладку «Objects». Натисніть «Upload». У вікні, що відкрилося, натисніть «Add files» та оберіть файл index.html

з вашого комп'ютера. Якщо є файл `style.css` - також додайте його через «Add files». Натисніть «Upload». Зробіть знімок екрана вкладки Objects з переліком завантажених файлів.

### **Крок 7. Відкриття та перевірка сайту**

Перейдіть на вкладку «Properties» вашого бакету. Прокрутіть донизу до розділу «Static website hosting». Знайдіть рядок «Bucket website endpoint» - це і є адреса вашого сайту. Вона виглядає приблизно так:

```
http://my-site-petrenko.s3-website.eu-central-1.amazonaws.com
```

Натисніть на це посилання або скопіюйте його та відкрийте у новій вкладці браузера. Сайт повинен відобразитися у браузері. Зробіть знімок екрана сайту у браузері з видимим URL у рядку адреси. Скопіюйте URL сайту та вставте у звіт.

### **Висновки до Варіанту 1**

Виконавши цю лабораторну роботу, студент навчився розміщувати статичний веб-сайт у хмарному сховищі Amazon S3. Зрозумів концепцію бакетів, об'єктів та Bucket Policy. Налаштував публічний доступ до файлів та режим статичного хостингу. Отримав досвід роботи з реальним хмарним сервісом корпоративного рівня, який обслуговує трильйони запитів щодня.

### **Варіанти завдань**

Варіант 2. Завантажити у S3-бакет повноцінний сайт-портфоліо з папкою images. Налаштувати кастомну сторінку помилки 404.

Варіант 3. Підключити до S3-сайту CloudFront (CDN) для прискорення завантаження. Порівняти час відгуку до і після підключення CDN.

Варіант 4. Налаштувати статичний хостинг у двох регіонах AWS (Frankfurt та Ireland) та порівняти час завантаження з різних географічних точок.

Варіант 5. Використати AWS CLI для завантаження файлів у S3 (команда `aws s3 sync`) замість веб-інтерфейсу.

Варіант 6. Налаштувати S3 Lifecycle Rules для автоматичного переміщення старих файлів до більш дешевого класу зберігання через 30 днів.

### **Корисні посилання та ресурси**

- AWS S3 User Guide - <https://docs.aws.amazon.com/s3/>
- AWS S3 Static Website Hosting -

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteHosting.html>

- AWS Free Tier - <https://aws.amazon.com/free>

# ЛАБОРАТОРНА РОБОТА № 8. ТЕМА: ЗНАЙОМСТВО З REST API. РОБОТА З ПУБЛІЧНИМИ API У POSTMAN

## Вступ до теми

API (Application Programming Interface) - це спосіб, яким програми спілкуються між собою. Розуміння REST API є однією з ключових навичок сучасного веб-розробника: більшість хмарних сервісів, включаючи AWS, Netlify, GitHub та тисячі інших платформ, керуються саме через REST API.

Реальний кейс. Компанія Stripe (платіжна платформа з оцінкою понад 50 мільярдів доларів) побудувала весь свій бізнес навколо якісного API. Їхня документація та дизайн API вважаються золотим стандартом у галузі. Slack, Twilio, SendGrid - всі ці компанії стали лідерами ринку завдяки тому, що розробники полюбили їх API. Це демонструє, що вміння проектувати та використовувати API є критичною навичкою.

## Мета роботи

Зрозуміти поняття API та REST. Навчитися тестувати публічні API за допомогою Postman, розуміти формат JSON та HTTP-статус-коди.

## Завдання до роботи

Студент встановлює Postman, виконує GET та POST запити до публічних API (НБУ та JSONPlaceholder), аналізує відповіді та статус-коди, зберігає колекцію запитів.

## Теоретичні відомості

### API та REST

API (Application Programming Interface) - це спосіб, яким програми спілкуються між собою. Аналогія: API - це меню у ресторані: воно показує, що можна замовити і в якій формі. REST API - найпоширеніший тип API для веб-сервісів. Він працює через HTTP (той самий протокол, що і браузер). Запит іде від клієнта до сервера, сервер відповідає даними у форматі JSON.

JSON (JavaScript Object Notation) - легкий формат обміну даними. Виглядає як текст з фігурними дужками:

```
{ "name": "Іван", "age": 20, "city": "Київ" }
```

| HTTP-метод | Для чого використовується |
|------------|---------------------------|
|------------|---------------------------|

|             |                                  |
|-------------|----------------------------------|
| GET         | Отримати дані (нічого не змінює) |
| POST        | Створити новий запис             |
| PUT / PATCH | Змінити існуючий запис           |
| DELETE      | Видалити запис                   |

| Код                       | Значення                |
|---------------------------|-------------------------|
| 200 OK                    | Запит виконано успішно  |
| 201 Created               | Новий ресурс створено   |
| 400 Bad Request           | Неправильний запит      |
| 401 Unauthorized          | Потрібна аутентифікація |
| 404 Not Found             | Ресурс не знайдено      |
| 500 Internal Server Error | Помилка на боці сервера |

## Детальне покрокове виконання - Варіант 1

### Крок 1. Завантаження та встановлення Postman

Відкрийте браузер та перейдіть: <https://www.postman.com/downloads/>. Натисніть кнопку «Download the App» для вашої операційної системи. Завантажте та встановіть програму. Запустіть Postman. На екрані входу натисніть «Skip and go to the app». Зробіть знімок екрана головного вікна Postman.

### Крок 2. Перший GET-запит до API НБУ

У Postman натисніть «+» щоб відкрити нову вкладку запити. Переконайтеся, що у випадяючому списку зліва від поля URL обрано метод «GET». У поле «Enter URL or paste text» введіть адресу:

<https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json>

Натисніть синю кнопку «Send». У нижній частині вікна з'явиться відповідь. Зверніть увагу на: «Status: 200 OK» (зелений) - запит успішний; «Time» - час відповіді в мілісекундах. У вкладці «Body» відображається JSON-масив з курсами валют. Знайдіть у відповіді об'єкти з полями «cc» (код валюти) та «rate» (курс). Запишіть у звіт курси для: USD (долар США), EUR (євро), GBP (британський фунт). Зробіть знімок екрана відповіді з видимим статусом 200 та даними валют.

### Крок 3. Збереження запиту

Натисніть «Save» (іконка дискети або Ctrl+S). У вікні збереження введіть ім'я запиту: «Курси НБУ». Натисніть «New Collection» - введіть назву: «JP9 - API тести» - «Create». Натисніть «Save». Колекція з'явиться у лівій панелі.

#### **Крок 4. Тестування JSONPlaceholder API - GET-запит**

Відкрийте нову вкладку запиту (натисніть «+»). Метод: GET. URL:

```
https://jsonplaceholder.typicode.com/posts/1
```

Натисніть «Send». Ви отримаєте JSON-об'єкт з одним постом. Запишіть у звіт значення полів: id, title, userId. Зробіть знімок екрана.

#### **Крок 5. POST-запит - створення запису**

Відкрийте нову вкладку. Змініть метод з GET на POST. URL:

```
https://jsonplaceholder.typicode.com/posts
```

Перейдіть на вкладку «Body» (під рядком URL). Оберіть «raw». У випадаючому списку праворуч змініть «Text» на «JSON». Вставте у текстове поле:

```
{
  "title": "Мій тестовий пост",
  "body": "Це текст мого першого POST-запиту",
  "userId": 1
}
```

Натисніть «Send». Перевірте: статус має бути «201 Created» (а не 200). У відповіді знайдіть поле «id» - JSONPlaceholder повернув ID нового запису. Зробіть знімок екрана з видимим статусом 201.

#### **Крок 6. Тест на неіснуючий ресурс**

Новий запит. Метод: GET. URL з неіснуючим ID:

```
https://jsonplaceholder.typicode.com/posts/9999
```

Натисніть «Send». Зверніть увагу: статус має бути «404 Not Found». Тіло відповіді - порожній об'єкт {}. Зробіть знімок екрана з видимим статусом 404. Поясніть у звіті: чому сервер повернув 404?

#### **Висновки до Варіанту 1**

Виконавши цю лабораторну роботу, студент практично освоїв основи роботи з REST API. Зрозумів структуру HTTP-запитів і відповідей, навчився розрізняти HTTP-методи GET, POST та статус-коди. Завдяки Postman перевірів реальний API Національного банку України та тренувальний API JSONPlaceholder. Набуті навички є фундаментом для роботи з хмарними API, включаючи API AWS, Netlify та GitHub.

#### **Варіанти завдань**

Варіант 2. Використати відкритий API OpenWeatherMap для отримання погоди у трьох українських містах. Порівняти отримані дані у таблиці.

Варіант 3. Протестувати GitHub REST API: отримати список публічних репозиторіїв вашого акаунту, деталі конкретного репозиторію та список комітів.

Варіант 4. Використати API PokeAPI для отримання даних про 5 покемонів. Порівняти їх характеристики у таблиці.

Варіант 5. Протестувати API JSONPlaceholder повністю: GET, POST, PUT, PATCH, DELETE. Зробити знімки екрана всіх запитів та відповідей.

Варіант 6. Написати JavaScript-код (fetch API) для отримання курсів НБУ та відображення їх на HTML-сторінці.

### **Корисні посилання та ресурси**

- Postman - офіційний сайт: <https://www.postman.com>
- JSONPlaceholder - тренувальний API:

<https://jsonplaceholder.typicode.com>

- API НБУ - <https://bank.gov.ua/ua/open-data/api-dev>
- MDN HTTP Overview -

<https://developer.mozilla.org/uk/docs/Web/HTTP/Overview>

# **ЛАБОРАТОРНА РОБОТА № 9. ТЕМА: ОСНОВИ КІБЕРБЕЗПЕКИ. ЗАХИСТ ОБЛІКОВИХ ЗАПИСІВ ТА ПЕРСОНАЛЬНИХ ДАНИХ В ІНТЕРНЕТІ**

## **Вступ до теми**

Кібербезпека є невід'ємною частиною роботи будь-якого ІТ-фахівця. Розробники несуть пряму відповідальність за безпеку даних своїх користувачів. Навички розпізнавання загроз та захисту облікових записів є критичними як для особистої безпеки, так і для захисту корпоративних систем.

Реальний кейс. У 2020 році компанія SolarWinds стала жертвою однієї з найбільших кібератак в історії. Зловмисники отримали доступ до систем через скомпрометований оновлений пакет програмного забезпечення, яке використовували понад 18 000 організацій, включаючи кілька федеральних агентств США. Причиною став, зокрема, слабкий пароль «solarwinds123», знайдений у відкритому доступі. Цей випадок показує, що навіть основи кібербезпеки можуть мати катастрофічні наслідки.

## **Мета роботи**

Сформувати практичні навички захисту власних облікових записів в Інтернеті. Навчитися оцінювати надійність паролів, налаштовувати двофакторну аутентифікацію, розпізнавати фішингові атаки та безпечно зберігати чутливі дані.

## **Завдання до роботи**

Студент аналізує надійність паролів через онлайн-інструмент, налаштовує двофакторну аутентифікацію для Google-акаунту, проходить тест на розпізнавання фішингу та заповнює таблицю ознак фішингових атак.

## **Теоретичні відомості**

### **Найпоширеніші кіберзагрози**

Фішинг (Phishing) - шахрайські листи або сайти, що імітують відомі сервіси (банки, Gmail, AWS) з метою викрасти логін і пароль. Ознаки: підозрілий домен, термінові вимоги, прохання «підтвердити акаунт».

Перебір паролів (Brute force) - автоматичний підбір паролів. Простий пароль «123456» зламується за секунди. Пароль з 12 символів різного регістру - за сотні років.

Витік даних (Data breach) - зловмисники отримують бази даних з паролями. Якщо ви використовуєте один пароль скрізь - після одного витоку всі ваші акаунти під загрозою.

Соціальна інженерія - маніпуляція людиною, а не технічна атака. «Я з технічної підтримки, дайте пароль» - жодна справжня компанія ніколи не просить пароль.

### Правила надійного пароля

| Слабкий пароль            | Надійний пароль                            |
|---------------------------|--------------------------------------------|
| 123456                    | k#9mP\$qL2@vN                              |
| password                  | Kov_basa!2025#drum                         |
| дата народження: 01012005 | Випадкова фраза:<br>«Кіт_їсть_піцу_o_3:00» |
| ім'я + рік: ivan2005      | Мінімум 12 символів, цифри,<br>спецсимволи |

### Двофакторна аутентифікація (2FA)

Двофакторна аутентифікація (2FA) - додатковий рівень захисту: навіть якщо зловмисник дізнався ваш пароль, без другого фактора (код із SMS або застосунку) він не зможе увійти. Рекомендований тип: застосунок-аутентифікатор (Google Authenticator, Microsoft Authenticator) - надійніший за SMS, бо SIM-картку можна підробити.

### Детальне покрокове виконання - Варіант 1

#### Крок 1. Аналіз надійності паролів

Відкрийте у браузері сайт: <https://www.security.org/how-secure-is-my-password/>

ВАЖЛИВО: ніколи не вводьте на таких сайтах свої реальні паролі! Використовуйте тестові варіанти схожої структури. Введіть по черзі 5 тестових паролів різної складності та запишіть результати у таблицю:

| Тестовий пароль | Час злому (за сайтом) | Оцінка |
|-----------------|-----------------------|--------|
| 123456          |                       |        |
| password2025    |                       |        |
| VasyI_2025!     |                       |        |

|                      |  |  |
|----------------------|--|--|
| K#9mP\$QL2@vN        |  |  |
| Ваш власний (схожий) |  |  |

Зробіть знімок екрана результату для найслабшого та найнадійнішого пароля.

## **Крок 2. Перевірка витоків даних**

Відкрийте сайт: <https://haveibeenpwned.com> - це офіційний сервіс перевірки, чи потрапив ваш email у відомі витoki баз даних. Введіть свою email-адресу. Якщо з'явилося повідомлення «Oh no - pwned!» - ваша email-адреса потрапила у витік. Запишіть, у яких витокax (якщо є). Зробіть знімок екрана результату перевірки (закрийте частину email на знімку для конфіденційності).

## **Крок 3. Встановлення Google Authenticator**

На вашому смартфоні відкрийте App Store (iOS) або Google Play (Android). Знайдіть та встановіть застосунок «Google Authenticator» від Google LLC.

## **Крок 4. Увімкнення 2FA для Google-акаунту**

Відкрийте у браузері: <https://myaccount.google.com/security>. Знайдіть розділ «Як ви входите в Google» - «Двоетапна перевірка» (2-Step Verification). Натисніть «Початок роботи» (Get started). Виконайте кроки майстра налаштування: підтвердьте пароль - додайте телефон - оберіть метод: «Застосунок Authenticator» - відскануйте QR-код через Google Authenticator на телефоні - введіть 6-значний код для підтвердження - увімкніть. Зробіть знімок екрана сторінки безпеки з написом «Двоетапна перевірка: Увімкнено».

## **Крок 5. Перевірка входу з 2FA**

Відкрийте вікно браузера в режимі «Інкогніто» (Ctrl+Shift+N). Спробуйте увійти до Google. Після введення пароля з'явиться запит коду з Authenticator. Відкрийте застосунок на телефоні, знайдіть код для Google (6 цифр, оновлюється кожні 30 секунд) та введіть його. Зробіть знімок екрана запиту коду підтвердження у браузері.

## **Крок 6. Тест на фішинг від Google**

Відкрийте: <https://phishingquiz.withgoogle.com> - це безкоштовний навчальний тест від Google, де вам показують реальні та фіктивні листи/сторінки. Пройдіть тест (8 запитань). Для кожного прикладу вирішіть: «Фішинг» чи «Легітимний лист». Зробіть знімок екрана фінального результату з вашим балом.

## **Крок 7. Аналіз ознак фішингу**

На основі тесту заповніть у звіті таблицю:

| Ознака фішингу | Приклад з тесту або реального життя |
|----------------|-------------------------------------|
|----------------|-------------------------------------|

|                                      |  |
|--------------------------------------|--|
| Підозрілий домен відправника         |  |
| Термінові вимоги (підтвердіть ЗАРАЗ) |  |
| Граматичні помилки в листі           |  |
| Підозрілі посилання                  |  |
| Прохання ввести пароль або картку    |  |

### **Висновки до Варіанту 1**

Виконавши цю лабораторну роботу, студент сформував практичні навички захисту власних облікових записів. Проаналізовано надійність паролів та зрозуміло, чому слабкі паролі є критичною загрозою. Налаштовано двофакторну аутентифікацію для Google-акаунту. Освоєно розпізнавання фішингових атак через тест Google. Зрозуміло, що кібербезпека є особистою відповідальністю кожного ІТ-фахівця: при розробці систем, що зберігають дані користувачів, необхідно дотримуватись принципів безпечного зберігання (хешування паролів, використання змінних середовища для ключів).

### **Варіанти завдань**

Варіант 2. Налаштувати менеджер паролів (Bitwarden або KeePass). Перенести щонайменше 5 різних паролів та показати безпечне генерування нових паролів.

Варіант 3. Дослідити реальний витік даних через haveibeenpwned.com. Проаналізувати три публічно відомі витіки та написати звіт про причини та наслідки.

Варіант 4. Налаштувати 2FA для трьох різних сервісів (Google, GitHub, Netlify) та продемонструвати вхід з кожним із них.

Варіант 5. Вивчити стандарт OWASP Top 10 та підготувати презентацію про три найпоширеніші веб-вразливості з прикладами.

Варіант 6. Провести аудит безпеки вашого GitHub-репозиторію: перевірити, чи немає відкритих API-ключів або паролів у коді, налаштувати GitHub Secret Scanning.

### **Корисні посилання та ресурси**

- Have I Been Pwned - перевірка витоків: <https://haveibeenpwned.com>
- Google Phishing Quiz - <https://phishingquiz.withgoogle.com>
- Google Account Security - <https://myaccount.google.com/security>
- OWASP Top 10 - <https://owasp.org/Top10/>

# ЛАБОРАТОРНА РОБОТА № 10. ТЕМА: ХМАРНЕ СХОВИЩЕ. ОРГАНІЗАЦІЯ ТА УПРАВЛІННЯ ФАЙЛАМИ В AMAZON S3

## Вступ до теми

Ця лабораторна робота поглиблює знання Amazon S3, отримані у попередніх роботах. Студент вивчає розширені можливості S3: організацію папок, класи зберігання, версіонування та метадані об'єктів. Ці навички необхідні для реальної роботи з хмарними сховищами у виробничому середовищі.

Реальний кейс. Spotify зберігає в Amazon S3 мільярди аудіофайлів. Вони використовують різні класи зберігання залежно від популярності пісні: нові треки та хіти - у S3 Standard для швидкого доступу; рідко прослуховуваний контент автоматично переміщується до S3 Standard-IA через 90 днів, а архівний контент - до S3 Glacier. Це дозволяє Spotify економити десятки мільйонів доларів щороку на зберіганні даних.

## Мета роботи

Навчитися ефективно управляти файлами в Amazon S3 через веб-консоль: організовувати папки, налаштовувати доступ, переглядати метадані та розуміти різні класи зберігання.

## Завдання до роботи

Студент організовує структуру папок у S3-бакеті, завантажує файли до різних папок, вивчає метадані об'єктів, вмикає версіонування та демонструє роботу з версіями файлів.

## Теоретичні відомості

### Класи зберігання S3

S3 зберігає файли як «об'єкти» у «бакетах». Технічно в S3 немає справжніх папок - лише об'єкти з іменами, що містять «/». Але консоль AWS відображає такі імена як папки для зручності.

| Клас зберігання | Для чого використовується                     | Орієнтовна вартість |
|-----------------|-----------------------------------------------|---------------------|
| S3 Standard     | Файли, до яких часто звертаються (сайти, API) | Найвища             |

|                    |                                                      |          |
|--------------------|------------------------------------------------------|----------|
| S3 Standard-IA     | Рідко потрібні файли (резервні копії старше місяця)  | Нижча    |
| S3 Glacier Instant | Архіви з миттєвим доступом (старі версії документів) | Найнижча |

### Версіонування S3

Версіонування (Versioning) - функція, яка зберігає всі версії кожного файлу. Якщо ви завантажили новий файл поверх старого - стара версія не видаляється, а зберігається. Це захист від випадкового перезапису. Lifecycle Rules (Правила життєвого циклу) - автоматизація управління файлами. Наприклад: «Через 30 днів перемістити до S3-IA, через 365 днів - до Glacier».

### Детальне покрокове виконання - Варіант 1

#### Крок 1. Відкриття та підготовка бакету

Увійдіть до AWS Console - S3. Відкрийте бакет my-site-[прізвище], створений у ЛР 8. Якщо такого бакету немає - створіть новий (назва: files-[прізвище], публічний доступ не потрібен для цієї роботи).

#### Крок 2. Створення папок

Перейдіть на вкладку «Objects». Натисніть «Create folder». У полі «Folder name» введіть: images. Натисніть «Create folder». Аналогічно створіть ще дві папки: documents та backups. Зробіть знімок екрана вкладки Objects з трьома папками.

#### Крок 3. Завантаження файлів у папки

Натисніть на папку images, щоб відкрити її. Натисніть «Upload» - «Add files». Оберіть будь-яке зображення (.jpg або .png) з вашого комп'ютера. Натисніть «Upload». Поверніться до кореня бакету та завантажте будь-який .txt файл у папку documents.

#### Крок 4. Перегляд Object URL та метаданих

Відкрийте папку images та натисніть на назву завантаженого файлу. Знайдіть рядок «Object URL» - це пряме посилання на файл. Перейдіть на вкладку «Properties» та заповніть у звіті таблицю метаданих:

| Поле метаданих             | Значення для вашого файлу |
|----------------------------|---------------------------|
| Object size (розмір)       |                           |
| Last modified (дата зміни) |                           |

|                                 |  |
|---------------------------------|--|
| Storage class (клас зберігання) |  |
| Content-Type (тип файлу)        |  |
| ETag (хеш файлу)                |  |

Зробіть знімок екрана вкладки Properties з видимими метаданими.

#### **Крок 5. Увімкнення версіонування**

Поверніться до головної сторінки бакету. Перейдіть на вкладку «Properties». Знайдіть розділ «Bucket Versioning». Натисніть «Edit» - оберіть «Enable» - «Save changes». Зробіть знімок екрана з підтвердженням «Bucket Versioning: Enabled».

#### **Крок 6. Завантаження першої версії файлу**

На комп'ютері відкрийте Блокнот, введіть: «Версія 1 - початкова». Збережіть як test.txt. Завантажте test.txt до бакету: «Upload» - «Add files» - test.txt - «Upload».

#### **Крок 7. Завантаження другої версії файлу**

НЕ змінюйте назву файлу. Відредагуйте test.txt - змініть текст на: «Версія 2 - оновлена». Збережіть. Знову завантажте цей самий файл test.txt до бакету. AWS S3 зберігає обидві версії.

#### **Крок 8. Перегляд версій**

У вкладці «Objects» знайдіть та увімкніть перемикач «Show versions». Тепер для файлу test.txt відображаються дві версії з різними «Version ID» та датами. Натисніть на стару версію - завантажте її - перевірте вміст (має бути «Версія 1»). Зробіть знімок екрана списку версій файлу.

### **Висновки до Варіанту 1**

Виконавши цю лабораторну роботу, студент поглибив знання Amazon S3 та освоїв його розширені можливості. Зорганізовано структуру бакету з папками, вивчено метадані об'єктів та класи зберігання. Практично продемонстровано роботу версіонування S3 - завантаження двох версій одного файлу та відновлення попередньої версії. Ці навички безпосередньо застосовуються у реальних хмарних проєктах для організації зберігання та захисту даних.

### **Варіанти завдань**

Варіант 2. Налаштувати Lifecycle Rule для S3-бакету: переміщення файлів до S3-IA через 30 днів та до Glacier через 90 днів. Задokumentувати налаштування скріншотами.

Варіант 3. Порівняти версіонування S3 та версіонування Google Drive. Написати детальний звіт з прикладами використання кожного.

Варіант 4. Налаштувати Cross-Region Replication (копіювання між регіонами) для S3-бакету між eu-central-1 та eu-west-1.

Варіант 5. Дослідити та налаштувати S3 Access Logs для відстеження всіх звернень до файлів бакету.

Варіант 6. Використати AWS S3 Select для виконання SQL-подібних запитів до CSV-файлу, завантаженого у S3-бакет.

### **Корисні посилання та ресурси**

- AWS S3 Storage Classes - <https://aws.amazon.com/s3/storage-classes/>
- AWS S3 Versioning -

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/Versioning.html>

- AWS S3 Lifecycle -

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lifecycle-mgmt.html>

# ЛАБОРАТОРНА РОБОТА № 12. ТЕМА: БЕЗСЕРВЕРНІ ОБЧИСЛЕННЯ.

## ПЕРША ФУНКЦІЯ AWS LAMBDA

### Вступ до теми

AWS Lambda є одним з найбільш революційних сервісів у хмарних обчисленнях. Концепція безсерверної архітектури змінила підхід до розробки та розгортання застосунків: замість підтримки серверів розробники зосереджуються виключно на бізнес-логіці. Lambda активно використовується у провідних технологічних компаніях.

Реальний кейс. Компанія Bustle (медіа-платформа з 50 мільйонами читачів на місяць) перенесла свій бекенд на AWS Lambda. До міграції вони витрачали 10 000 доларів на місяць на сервери EC2. Після переходу на Lambda їх витрати знизились до 400 доларів на місяць - скорочення на 96%. Водночас час завантаження сторінок покращився на 40%. Це типовий приклад того, як Lambda дозволяє платити лише за реальне використання.

### Мета роботи

Ознайомитися з концепцією Serverless (безсерверної) архітектури. Створити та запустити першу Lambda-функцію на AWS. Зрозуміти поняття «подія», «контекст», «функція-обробник».

### Завдання до роботи

Студент створює Lambda-функцію на Python, тестує її з різними вхідними даними, переглядає логи у CloudWatch та аналізує метрики виконання.

### Теоретичні відомості

#### Безсерверна архітектура

Serverless (безсерверна архітектура) - модель, де розробник пише лише код функції, а хмарний провайдер (AWS) сам турбується про сервери, їх запуск і масштабування. «Без серверів» не означає, що серверів немає - просто ви про них не думаєте. Аналогія: звичайний сервер - як власна кухня (платиш завжди, навіть коли не готуєш). Lambda - як замовлення їжі: платиш лише тоді, коли готується страва.

| Критерій     | Традиційний сервер | AWS Lambda                    |
|--------------|--------------------|-------------------------------|
| Коли працює? | Постійно (24/7)    | Лише під час виконання запиту |

|               |                    |                                        |
|---------------|--------------------|----------------------------------------|
| Оплата        | За весь час роботи | Лише за час виконання<br>(мілісекунди) |
| Масштабування | Вручну             | Автоматично та миттєво                 |
| Free Tier     | Обмежений          | 1 000 000 запитів/міс<br>безкоштовно   |

## Детальне покрокове виконання - Варіант 1

### Крок 1. Відкриття сервісу Lambda

Увійдіть до AWS Console. У рядку пошуку введіть «Lambda» та натисніть на результат «Lambda - Run Code without Thinking about Servers». Переконайтеся, що у правому верхньому куті обрано регіон EU (Frankfurt). Натисніть кнопку «Create function» (Створити функцію).

### Крок 2. Налаштування функції

На сторінці «Create function» оберіть: «Author from scratch» (З нуля). Заповніть поля: Function name: hello-student; Runtime: Python 3.12; Architecture: x86\_64. Всі інші налаштування залиште за замовчуванням. Натисніть «Create function». Зачекайте 10-20 секунд. Зробіть знімок екрана сторінки щойно створеної функції.

### Крок 3. Введення коду функції

На сторінці функції перейдіть на вкладку «Code» (Код). У редакторі коду видаліть весь код (Ctrl+A - Delete) та вставте наступний код:

```
import json
import datetime

def lambda_handler(event, context):
    # event - вхідні дані (словник Python)
    # context - інформація про виконання функції

    # Зчитуємо ім'я з вхідних даних або ставимо за замовчуванням
    name = event.get("name", "Студент")

    # Отримуємо поточний час
    now = datetime.datetime.now().strftime("%H:%M:%S")

    # Повертаємо результат - словник стає JSON
    return {
        "statusCode": 200,
        "message": f"Привіт, {name}! Поточний час: {now}",
        "success": True
    }
```

}

Натисніть кнопку «Deploy» (Розгорнути) - вона знаходиться над редактором коду. Зачекайте секунду, поки кнопка знову стане активною - це означає, що код збережено та розгорнуто.

#### **Крок 4. Створення тест-події та тестування**

На вкладці «Code» натисніть кнопку «Test» (або перейдіть на вкладку «Test»). З'явиться вікно «Configure test event». Оберіть «Create new event». У полі «Event name» введіть: test-with-name. Видаліть вміст поля «Event JSON» та вставте:

```
{ "name": "Ваше Ім'я" }
```

(Замініть «Ваше Ім'я» на своє реальне ім'я). Натисніть «Save». Натисніть кнопку «Test» ще раз. Якщо бачите зелений напис «Execution result: succeeded» - функція запрацювала. У розділі «Response» знайдіть поле «message» - там має бути «Привіт, [ваше ім'я]! Поточний час: XX:XX:XX». Зробіть знімок екрана успішного виконання.

#### **Крок 5. Тест без імені**

Натисніть стрілку поруч із кнопкою «Test» - «Configure test event». Змініть Event JSON на порожній об'єкт: {}. Натисніть «Save» - «Test». Тепер у відповіді замість вашого імені має з'явитись «Студент» (значення за замовчуванням). Зробіть знімок екрана. У звіті поясніть: чому функція повернула «Студент»?

#### **Крок 6. Перегляд логів у CloudWatch**

Перейдіть на вкладку «Monitor» на сторінці функції. Ви побачите графіки: Invocations (кількість викликів), Duration (час виконання), Errors (помилки). Натисніть «View logs in CloudWatch». На сторінці CloudWatch знайдіть «Log streams». Натисніть на найновіший (верхній) рядок. Серед записів знайдіть рядок, що починається з «REPORT». Він виглядає приблизно так:

```
REPORT RequestId: ... Duration: 2.45 ms Billed Duration: 3 ms  
Memory Size: 128 MB Max Memory Used: 35 MB
```

Запишіть у звіт значення: Duration (реальний час виконання), Billed Duration (час, за який нараховується оплата), Max Memory Used (використана пам'ять). Зробіть знімок екрана рядка REPORT.

#### **Висновки до Варіанту 1**

Виконавши цю лабораторну роботу, студент практично ознайомився з концепцією Serverless-архітектури та сервісом AWS Lambda. Створено першу Lambda-функцію на Python, протестовано її роботу з різними вхідними даними та проаналізовано метрики виконання у CloudWatch. Зрозуміло, що безсерверна архітектура дозволяє платити лише за

реальне використання ресурсів і забезпечує автоматичне масштабування, що робить її ефективним рішенням для багатьох задач у сучасній веб-розробці.

### **Варіанти завдань**

Варіант 2. Модифікувати Lambda-функцію: додати обробку масиву чисел з вхідного event та повертати суму, мінімум, максимум і середнє значення.

Варіант 3. Створити Lambda-функцію, яка зчитує файл з S3-бакету та повертає його вміст у відповіді.

Варіант 4. Налаштувати Lambda-функцію з тригером від API Gateway та перевірити виклик функції через HTTP-запит у Postman.

Варіант 5. Створити Lambda-функцію, яка записує повідомлення у DynamoDB при кожному виклику.

Варіант 6. Налаштувати Lambda-функцію з тригером від S3: при завантаженні нового файлу до бакету функція автоматично виконується та логує ім'я файлу у CloudWatch.

### **Корисні посилання та ресурси**

- AWS Lambda Developer Guide - <https://docs.aws.amazon.com/lambda/latest/dg/>
- AWS CloudWatch - <https://docs.aws.amazon.com/cloudwatch/>
- AWS Lambda Pricing - <https://aws.amazon.com/lambda/pricing/>
- Python 3 Documentation - <https://docs.python.org/3/>

## ЛАБОРАТОРНА РОБОТА № 12

### ТЕМА: КОМАНДНИЙ РЯДОК ДЛЯ ХМАРИ. УПРАВЛІННЯ S3 ЧЕРЕЗ AWS CLI

#### Вступ до теми

AWS CLI (Command Line Interface) є стандартним інструментом для роботи з хмарними ресурсами у виробничому середовищі. Більшість реальних проєктів використовують CLI для автоматизації, скриптування та інтеграції з CI/CD pipelines. Вміння ефективно працювати з CLI суттєво підвищує продуктивність та відкриває можливості для автоматизації рутинних задач.

Реальний кейс. Велика українська e-commerce компанія автоматизувала за допомогою AWS CLI щоденне резервне копіювання 500 ГБ даних до S3. Сценарій виглядає так: вночі запускається bash-скрипт, який використовує команду `aws s3 sync` для синхронізації бази даних та медіафайлів з S3. Цей скрипт займає 15 рядків коду та замінює роботу, яка раніше вимагала ручного втручання інженера щодня. Вартість зберігання у S3 виявилась на 60% нижчою за альтернативні рішення.

#### Мета роботи

Встановити та налаштувати AWS CLI. Навчитися виконувати операції з файлами в Amazon S3 через командний рядок і зрозуміти переваги CLI перед веб-консоллю.

#### Завдання до роботи

Студент встановлює AWS CLI, налаштовує облікові дані через Access Key, виконує базові операції з S3 (перегляд, завантаження, скачування, синхронізація, видалення) та документує результати скріншотами.

#### Теоретичні відомості

##### 13.1 AWS CLI - переваги та структура

AWS CLI (Command Line Interface) - офіційний інструмент для управління ресурсами AWS через термінал. Він дозволяє робити все те саме, що і веб-консоль, але однією командою. Навіщо CLI замість браузера: одна команда замість 5-10 кліків; можливість автоматизації через скрипти; зручно при роботі з великою кількістю файлів.

Структура команди AWS CLI:

```
aws [сервіс] [операція] [параметри]
aws s3 cp файл.txt s3://мій-бакет/
```

| Команда | Що робить |
|---------|-----------|
|---------|-----------|

|                                  |                                         |
|----------------------------------|-----------------------------------------|
| aws s3 ls                        | Показати всі ваші бакети                |
| aws s3 ls s3://назва-бакету/     | Показати файли у бакеті                 |
| aws s3 cp файл.txt s3://бакет/   | Завантажити файл до S3                  |
| aws s3 cp s3://бакет/файл.txt ./ | Скачати файл з S3                       |
| aws s3 sync ./папка s3://бакет/  | Синхронізувати локальну папку з бакетом |
| aws s3 rm s3://бакет/файл.txt    | Видалити файл з S3                      |

## Детальне покрокове виконання - Варіант 1

### Крок 1. Завантаження та встановлення AWS CLI

Відкрийте браузер та перейдіть: <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>. Знайдіть розділ для вашої операційної системи. Windows: завантажте файл «AWSCLIV2.msi» (64-bit), запустіть інсталятор та дотримуйтесь стандартних кроків. macOS: завантажте файл «AWSCLIV2.pkg» та запустіть.

### Крок 2. Перевірка встановлення

ВАЖЛИВО: після встановлення закрийте всі відкриті вікна терміналу та відкрийте NOBE вікно. Windows: натисніть Win+R - введіть cmd - Enter. macOS: відкрийте Terminal. У новому терміналі введіть команду:

```
aws --version
```

Ви повинні побачити щось подібне: aws-cli/2.x.x Python/3.x.x Windows/10 exe/AMD64. Зробіть знімок екрана терміналу з результатом команди.

### Крок 3. Створення Access Key у AWS Console

Відкрийте AWS Console у браузері. Перейдіть: IAM - Users - student-user - вкладка «Security credentials». Знайдіть розділ «Access keys» та натисніть «Create access key». Оберіть «Local code» та натисніть «Next». Натисніть «Create access key». З'явиться сторінка з двома значеннями. ОДРАЗУ скопіюйте обидва та збережіть у текстовий файл: Access Key ID (починається з AKIA...) та Secret Access Key (довгий рядок з букв та цифр). Натисніть «Done».

КРИТИЧНО ВАЖЛИВО: Secret Access Key показується ЛИШЕ ОДИН РАЗ. Якщо закрити сторінку, не зберігши - доведеться видалити цей ключ та створити новий. Ніколи не публікуйте Access Key у коді, GitHub або будь-де публічно - це дає доступ до вашого AWS-акаунту!

### Крок 4. Налаштування AWS CLI через aws configure

Поверніться до терміналу. Введіть команду:

```
aws configure
```

CLI задасть чотири запитання по черзі. Відповідайте так:

```
AWS Access Key ID [None]: вставте ваш Access Key ID
```

```
AWS Secret Access Key [None]: вставте ваш Secret Access Key
```

```
Default region name [None]: eu-central-1
```

```
Default output format [None]: json
```

Після введення кожного значення натискайте Enter. Після останнього Enter конфігурація збережеться.

### Крок 5. Перевірка підключення

Введіть команду перевірки:

```
aws sts get-caller-identity
```

Якщо конфігурація вірна - побачите JSON з вашими даними: UserId, Account (номер акаунту), Arn. Зробіть знімок екрана результату.

### Крок 6. Перегляд бакетів та їх вмісту

Введіть команду для перегляду всіх ваших бакетів:

```
aws s3 ls
```

Ви побачите список бакетів з датою створення та назвою. Знайдіть у списку бакет my-site-[прізвище]. Зробіть знімок екрана. Введіть команду для перегляду файлів у вашому бакеті (замініть назву):

```
aws s3 ls s3://my-site-[прізвище]/
```

Зробіть знімок екрана.

### Крок 7. Завантаження файлу до S3

Спочатку створіть тестовий файл. У терміналі (Windows):

```
echo Тестовий файл для AWS CLI > cli-test.txt
```

Або на macOS/Linux:

```
echo "Тестовий файл для AWS CLI" > cli-test.txt
```

Тепер завантажте його до S3:

```
aws s3 cp cli-test.txt s3://my-site-[прізвище]/cli-test.txt
```

Перевірте, що файл з'явився у бакеті:

```
aws s3 ls s3://my-site-[прізвище]/
```

Зробіть знімок екрана команди cp та результату перевірки ls.

### Крок 8. Скачування файлу з S3

Завантажте щойно завантажений файл назад на свій комп'ютер до нової папки:

```
mkdir downloaded
```

```
aws s3 cp s3://my-site-[прізвище]/cli-test.txt ./downloaded/cli-test.txt
```

Перевірте, що файл завантажився: він з'явиться у папці downloaded. Зробіть знімок екрана.

### Крок 9. Синхронізація папки

Команда `sync` оновлює лише ті файли, що змінилися. Введіть (замінивши шлях на реальний до папки `my-portfolio`):

```
aws s3 sync ./my-portfolio s3://my-site-[прізвище]/
```

CLI виведе список файлів, які були завантажені або оновлені. Зробіть знімок екрана.

### Крок 10. Видалення файлу

Видаліть тестовий файл з бакету:

```
aws s3 rm s3://my-site-[прізвище]/cli-test.txt
```

Перевірте, що файл зник:

```
aws s3 ls s3://my-site-[прізвище]/
```

Зробіть знімок екрана команди `rm` та результату перевірки.

### Висновки до Варіанту 1

Виконавши цю лабораторну роботу, студент встановив та налаштував AWS CLI - офіційний інструмент командного рядка для роботи з хмарними ресурсами AWS. Практично виконано усі основні операції з файлами в Amazon S3: перегляд бакетів, завантаження файлів, скачування, синхронізація папки та видалення. Освоєно концепцію Access Key та зрозуміло важливість захисту облікових даних CLI. Набуті навички дозволяють автоматизувати рутинні операції зі сховищем через скрипти та CI/CD pipelines.

### Варіанти завдань

Варіант 2. Написати `bash`-скрипт (Windows batch або PowerShell), який автоматично синхронізує локальну папку з S3-бакетом та виводить у лог кількість синхронізованих файлів.

Варіант 3. Використати AWS CLI для перелічення всіх Lambda-функцій в акаунті та отримання детальної інформації про функцію `hello-student` з ЛР 12.

Варіант 4. Скласти та виконати послідовність команд AWS CLI для повного розгортання статичного сайту на S3: створення бакету, налаштування хостингу, завантаження файлів, налаштування Bucket Policy.

Варіант 5. Порівняти швидкість виконання операцій через веб-консоль AWS та через CLI (завантаження 10 файлів). Написати звіт з вимірами часу.

Варіант 6. Налаштувати декілька AWS CLI профілів (для різних акаунтів або регіонів) та продемонструвати їх використання через параметр `--profile`.

### Корисні посилання та ресурси

- AWS CLI Installation Guide - <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>
- AWS CLI S3 Commands - <https://docs.aws.amazon.com/cli/latest/reference/s3/>
- AWS CLI Configuration - <https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html>

- AWS CLI Cheat Sheet - <https://www.blumatador.com/learn/aws-cli-cheatsheet>

## Література

1. Проектування інформаційних систем: методичні вказівки /Уклад. О.В.Ізмайлова , – Київ : КНУБА , 2023 . – 28 с . <https://repository.knuba.edu.ua/items/f1ee0cf1-e830-43a7-b33e-dc2af6f40378>
2. Ніколюк П. К. Моделювання систем : навч. посіб. для здобувачів вищої освіти спеціальності 122 Комп'ютерні науки. Вінниця: ДонНУ, 2023. 228 с.
3. Міца О.В., Лавер В.О. Системний аналіз : навч.-метод. посіб. / О.В. Міца, В.О. Лавер. Ужгород : вид-во ПП «АУТДОР - ШАРК», 2021. 63 с.
4. Ізмайлова, О. В. Проектування інформаційних систем: навч. посібник : для студ. галузі знань 12 "Інформаційні технології" - Київ : КНУБА, 2022. - 87 с.
5. Литвин В. В., Пасічник В. В., Шаховська Н. Б. Проектування інформаційних систем: Навчальний посібник. Львів: «Магнолія-2006», 2021. 380 с.
6. Новіков Ф. В. Техніко-економічне обґрунтування сучасних технологій виробництва : навч. посібник / Ф. В. Новіков, Д. Ф. Новіков, О. А. Єрмоленко та ін. – Дніпро: ЛІРА, 2022. – 256 с. Режим доступу: <http://www.repository.hneu.edu.ua/handle/123456789/28117>
7. Сучасні інформаційні технології та системи [Електронний ресурс] : монографія / Н. Г. Аксак, Л. Е. Гризун, О. В. Щербаков [та ін.] ; за заг.ред. Пономаренка В. С. – Харків : ХНЕУ ім. С. Кузнеця, 2022. – 270 с. Режим доступу: <http://repository.hneu.edu.ua/handle/123456789/29233>
8. Теорія систем та системний аналіз : курс лекцій / В. В. Тютюнник, О. О. Писклакова. – Харків : Друкарня Мадрид, 2020. – 108 с.
9. Рубан І. В. Особливості створення системи підтримки прийняття антикризових рішень в умовах невизначеності вхідної інформації при надзвичайних ситуаціях / І. В. Рубан, В. В. Тютюнник, О. О. Тютюнник // Сучасні інформаційні технології у сфері безпеки та оборони. – 2021. – №1(40). – С. 75-84. Режим доступу: <http://repository.hneu.edu.ua/handle/123456789/25600>
10. Виклюк Я. І. Моделювання складних систем : посібник / Я. І. Виклюк, Р.М. Камінський, В. В. Пасічник. – Львів : Вид-во «Новий Світ –2000», 2020. – 404 с., 2023. – 39 с.
11. Зелінська О. В. Інформаційні системи та технології в галузі : навч. посіб. / О. В. Зелінська, Н. А. Потапова, Л. О. Волонтир. – Вінниця : ВНАУ, 2020. – 253 с
12. Крєневич А. П. Алгоритми і структури даних : підручник / А. П. Крєневич. – Київ : ВПЦ «Київський Університет», 2021. – 200 с

13. Методологія інформаційних систем та баз даних: теоретичний і практичний підходи : навч. посіб. / уклад. Ю. О. Ушенко, М. Л. Ковальчук, М. С. Гавриляк, А. Л. Негрич. – Чернівці : Чернівецький нац. ун-т ім. Ю. Федьковича, 2021. – 240 с.
14. Мінухін С. Метод збільшення продуктивності Apache Spark на основі сегментування даних і налаштувань конфігураційних параметрів [Електронний ресурс] / С. Мінухін, Н. Коптілов // Сучасний стан наукових досліджень та технологій промисловості. – 2024. – № 1(27). – С. 128–139. – Режим доступу : <https://doi.org/10.30837/ITSSI.2024.27.128>
15. Kief Morris. Infrastructure as Code. O'Reilly Media, 2020 – 462 p. Michael Kreef, Mitchell Hashimoto. Terraform Cookbook: Efficiently Define, Launch, and Manage Infrastructure as Code Across Various Cloud Platforms. Packt Publishing, 2020 – 366 p.
16. Thomas Kurian Tekanat. Datadog Cloud Monitoring Quick Start Guide. Packt Publishing, 2021 – 318 p.
17. Brendan Burns, Joe Beda, Kelsey Hightower, Lachlan Evenson. Kubernetes: Up and Running, 3rd Edition. O'Reilly Media, 2022 – 540 p.
18. Raphael Hertzog, Jim O’Gorman, Mati Aharoni, Joe O’Gorman. Kali Linux Revealed: Mastering the Penetration Testing Distribution. OffSec Press, 2021 – 346 p.

#### **Інтернет-ресурси:**

1. Академія web розробки MDN [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/>
2. HTML Tutorial [Електронний ресурс]. – Режим доступу: <https://www.tutorialspoint.com/html/index.htm>
3. CSS Tutorial [Електронний ресурс]. – Режим доступу: <https://www.tutorialspoint.com/css/index.htm>
4. Javascript Tutorial [Електронний ресурс]. – Режим доступу: <https://www.tutorialspoint.com/javascript/index.htm>
5. SQL Tutorial [Електронний ресурс]. – Режим доступу: <https://www.tutorialspoint.com/sql/index.htm>
6. Навчальна платформа Coursera, Secure Software Design [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.coursera.org/programs/program-natsional-nii-tiekhnichnii-univiersitiet-ukrayini-kiyivs-kii/specializations/secure-software-design>
7. Навчальна платформа Coursera, Information Systems [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.coursera.org/programs/program-natsional-nii-tiekhnichnii-univiersitiet-ukrayini-kiyivs-kii/specializations/information-systems>
8. Навчальна платформа Udey, Software Architecture & System Design Practical Case Studies [Електронний ресурс]. – Режим доступу до ресурсу: <https://ua.udemy.com/course/software-architecture-system-design-practical-case-studies>

9. Навчальна платформа Udemy, Software Architecture & Design of Modern Large Scale Systems [Електронний ресурс]. – Режим доступу до ресурсу: <https://ua.udemy.com/course/software-architecture-design-of-modern-large-scale-systems/>

10. Навчальна платформа Udemy, Software Architecture & Technology of Large-Scale Systems [Електронний ресурс]. – Режим доступу до ресурсу: <https://ua.udemy.com/course/developer-to-architect/>

11. Навчальна платформа Wintellectnow, Architecting Distributed Cloud Applications [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.wintellectnow.com/Home/SeriesDetail?seriesId=architecting-distributed-cloud-applications>

### **Інформаційні ресурси**

1. Електронна бібліотека ЛьвДУВС: URL:<https://www1.lvduvs.edu.ua/elektronnyi-kataloh>

2. Державна науково-технічна бібліотека України. URL: <http://www.gntb.n-t.org>.

Сайт національної бібліотеки України імені В.І.Вернадського. URL: <http://www.nbuv.gov.ua>.

3. Бібліотека ПДАУ. URL: <https://www.pdaa.edu.ua/content/biblioteka>.

4. Atlassian solutions. URL: <https://www.atlassian.com/>

5. Bitbucket - The code collaboration platform for modern software teams. URL:<https://bitbucket.org/>

6. Docker. URL: <https://www.docker.com/>

7. drawSQL - Design, visualize and collaborate on entity relationship diagrams for your databases. URL:

<https://drawsql.app/>

8. git – fast version control. URL: <https://git-scm.com/>

9. GitHub - Where the world builds software. URL: <https://github.com/>

10. GitLab – the DevOps Platform. URL: <https://about.gitlab.com/>

11. Kubernetes. URL: <https://kubernetes.io/>

12. Trello. URL: [ps://trello.com/home](https://trello.com/home)

13. Visual Studio Code. URL: <https://code.visualstudio.com/>