# ЗМІСТ

1. ВСТУП ……………………………………………………………………… 3
2. ОПИС ПРОЄКТУ ………………………………………………………….. 4
3. РЕАЛІЗОВАНІ ТЕМИ ПРАКТИКИ …………………………………...…. 6

3.1 Створення та налаштування WEB-проєкту …………………… 6

3.2 Робота з DOM-деревом ……………….………………………… 8

3.3 Обробка подій користувача …………..………………………... 10

3.4 Робота з масивами та об’єктами ………………………………. 12

3.5 Розробка HTML-форм та валідація ……..…………………….. 14

3.6 Використання Bootstrap та Axios ……..………………………. 16

3.7 Взаємодія з зовнішніми API …………..………………………. 18

3.8 Обробка помилок у JavaScript ………..……………………….. 20

3.9 Пагінація та нескінченний скрол …….……………………….. 22

3.10 Використання localStorage ………….……………………….. 24

1. ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ ………………..………………….. 26
2. СТРУКТУРА ПРОЄКТУ ……………………………………………….... 28
3. ІНСТРУКЦІЯ З ВИКОРИСТАННЯ ……………….…………………… 30
4. ВИСНОВКИ ……………………………………………………………… 32
5. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ………………………………... 33

**1.** **ВСТУП**

Метою навчальної практики є закріплення та поглиблення знань, отриманих під час теоретичного навчання, застосування необхідних вмінь та навичок у розробці веб-застосунків.

В рамках практики було обрано Варіант 6: Photo Gallery – веб-застосунок для відображення добірки фотографій з Unsplash API із можливістю перегляду в повному розмірі.

## Завдання практики

Основними завданнями навчальної практики є:

* Оволодіння сучасними методами та інструментами розробки вебзастосунків, зокрема роботою з HTML, CSS, JavaScript та системами контролю версій.
* Закріплення та розвиток знань і вмінь, отриманих студентами під час теоретичного навчання, шляхом виконання практичного проєкту.
* Набуття навичок проєктування, створення та тестування клієнтських вебзастосунків.
* Виховання відповідальності за результати своєї роботи та вміння працювати самостійно, використовуючи системи керування проєктами.
* Формування цифрової культури праці, дбайливого ставлення до ресурсів та дотримання правил кібербезпеки.

## Обраний варіант

Варіант 6: Photo Gallery – відображати добірку фотографій з API (Unsplash) із можливістю перегляду в повному розмірі.

# 2. ОПИС ПРОЄКТУ

Photo Gallery – це інтерактивний веб-застосунок для пошуку та перегляду фотографій з безкоштовного API Unsplash.

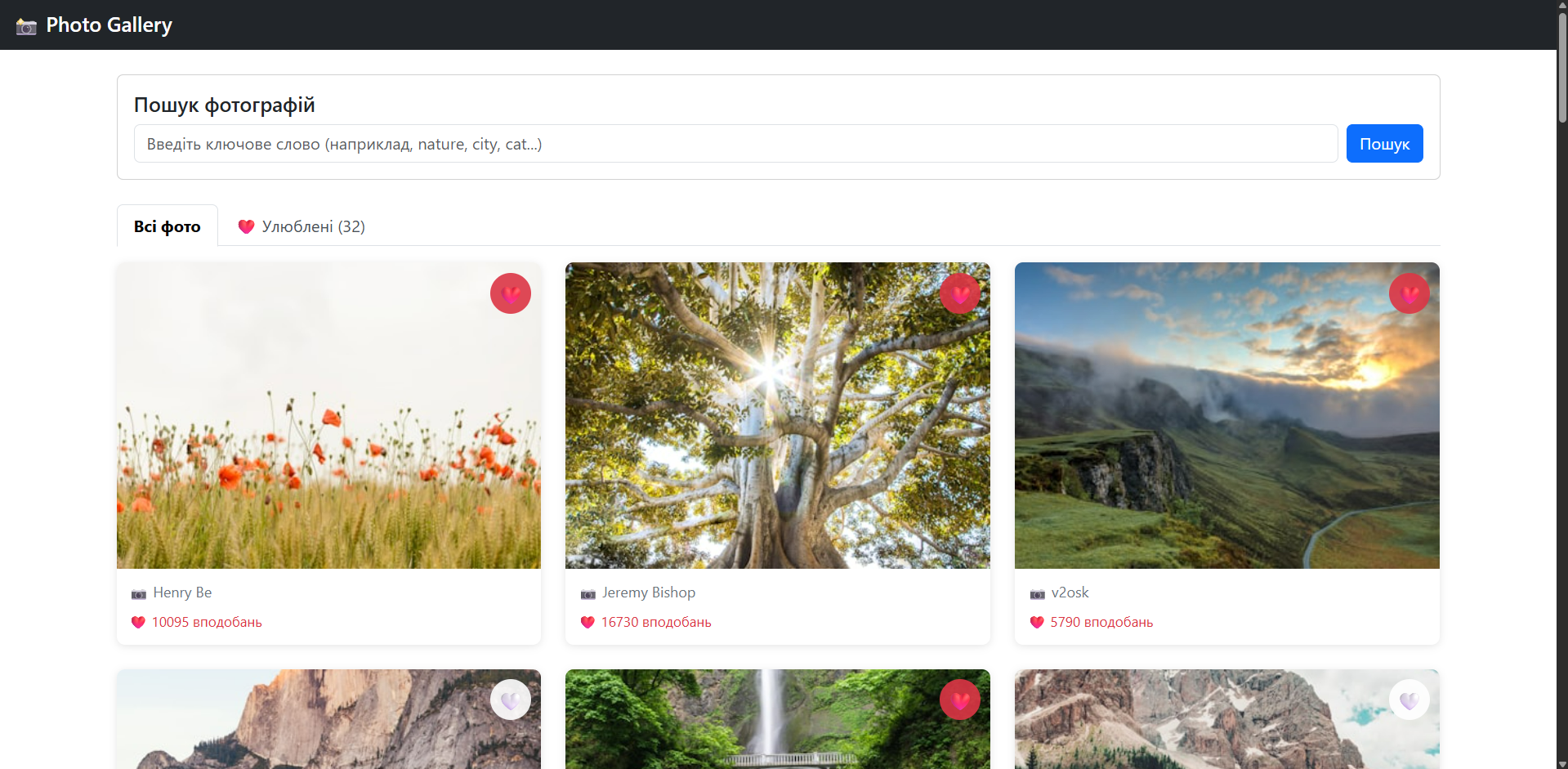
## Основні характеристики

* Тип застосунку: Single Page Application (SPA)
* API: Unsplash API
* Збирач модулів: Vite 7.2.4
* Фреймворк UI: Bootstrap 5.3.8
* HTTP клієнт: Axios 1.13.2
* Мова програмування: JavaScript ES6+
* Система контролю версій: Git

## Функціональність застосунку

1. Пошук фотографій за ключовими словами через форму пошуку.
2. Відображення результатів у вигляді адаптивної галереї з картками фотографій.
3. Перегляд у повному розмірі – модальне вікно з детальною інформацією про фото.
4. Збереження улюблених – можливість додавати фото в улюблені з використанням localStorage.
5. Нескінченний скрол – автоматичне завантаження додаткових фотографій при прокручуванні сторінки.
6. Обробка помилок – детальні повідомлення про помилки API, мережі та валідації.

## Головна сторінка застосунку



# 3. РЕАЛІЗОВАНІ ТЕМИ ПРАКТИКИ

## 

## 3.1 Створення та налаштування WEB-проєкту

Проєкт було створено з використанням сучасних інструментів розробки.

### Ініціалізація проєкту з Vite

Для створення проєкту використано команду:

npm create vite@latest photo-gallery -- --template vanilla

### Структура package.json

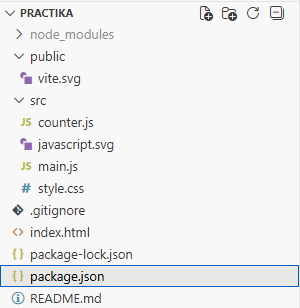
Файл конфігурації проєкту містить всі необхідні залежності:

{  
 "name": "photo-gallery",  
 "private": true,  
 "version": "0.0.0",  
 "type": "module",  
 "scripts": {  
 "dev": "vite",  
 "build": "vite build",  
 "preview": "vite preview"  
 },  
 "devDependencies": {  
 "vite": "^7.2.4"  
 },  
 "dependencies": {  
 "axios": "^1.13.2",  
 "bootstrap": "^5.3.8"  
 }  
}

### Git репозиторій

Проєкт розміщено в Git репозиторії з правильно налаштованим .gitignore:

node\_modules  
dist  
dist-ssr  
\*.local



### Базова HTML структура

Створено семантичну HTML структуру з підключенням скриптів:

<!doctype html>  
<html lang="uk">  
 <head>  
 <meta charset="UTF-8" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <title>Photo Gallery - Галерея Фотографій</title>  
 </head>  
 <body>  
 <div class="container my-4">  
 <!-- Контент -->  
 </div>

<script type="module" src="/src/main.js"></script>  
 </body>  
</html>

## 

## 3.2 Робота з DOM-деревом

Реалізовано повноцінну роботу з DOM для динамічного створення та управління елементами галереї.

### Вибір елементів

Використано різні методи вибору елементів:

// Вибір за ID  
const searchForm = document.getElementById('searchForm');  
const photoGallery = document.getElementById('photoGallery');  
// Вибір за селектором  
const modal = document.getElementById('photoModal');

### Динамічне створення елементів

Функція створення картки фотографії (src/main.js, рядки 147-167):

function createPhotoCard(photo) {  
 const col = document.createElement('div');  
 col.className = 'col';  
 const isFav = isFavorite(photo.id);  
 col.innerHTML = `  
 <div class="photo-card" data-photo-id="${photo.id}">  
 <button class="favorite-btn ${isFav ? 'favorited' : ''}"  
 data-photo-id="${photo.id}">

${isFav ? '❤️' : '🤍'}

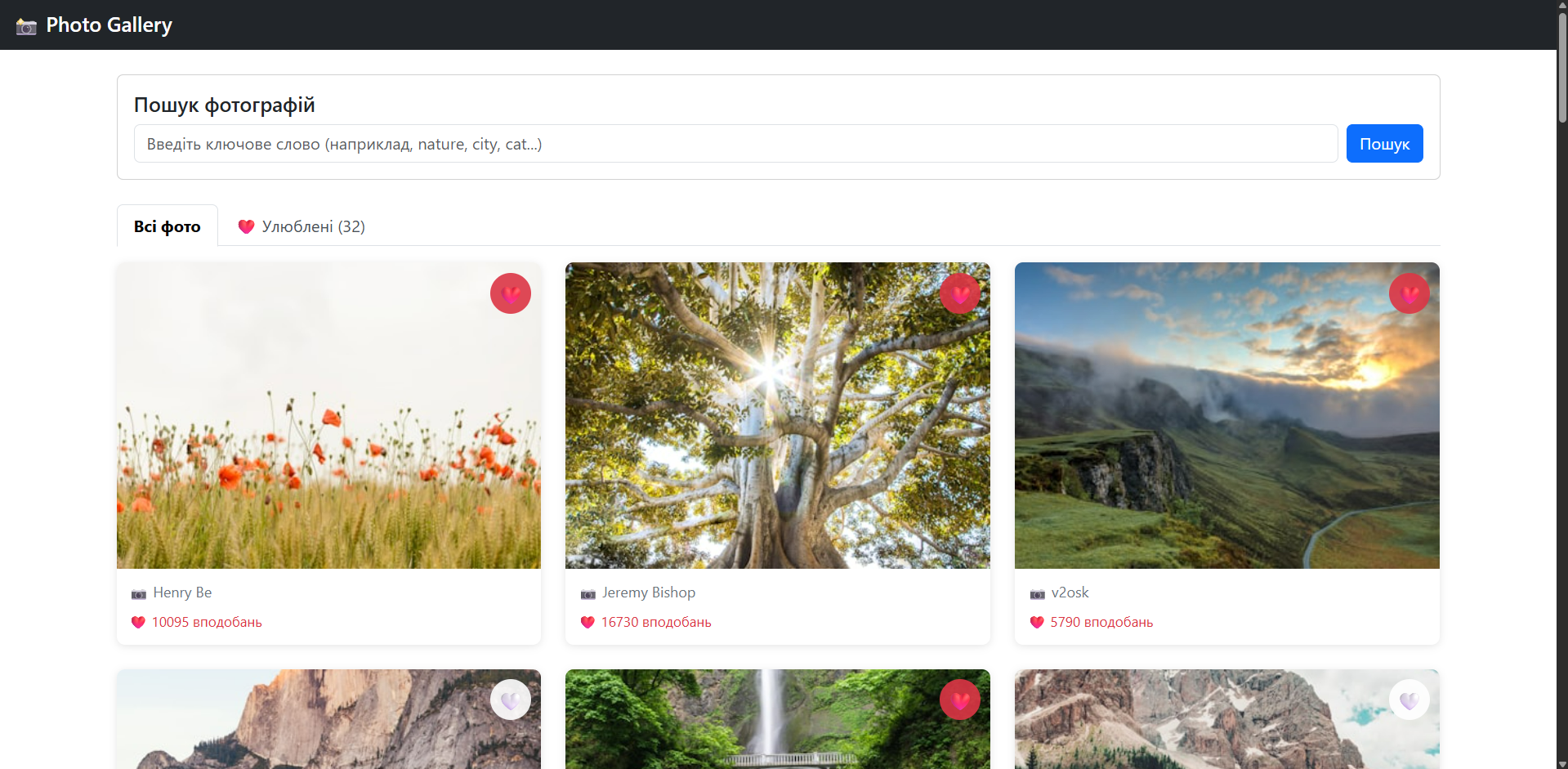
</button>  
 <img src="${photo.urls.small}"

alt="${photo.alt\_description || 'Photo'}"  
 loading="lazy" />  
 <div class="photo-info">  
 <div class="photo-author">📷 ${photo.user.name}</div>  
 <div class="photo-likes">❤️ ${photo.likes} вподобань</div>  
 </div>  
 </div>  
 `;  
 return col;  
}

### Додавання елементів до DOM

Відображення фотографій у галереї (src/main.js, рядки 192-193):

photos.forEach(photo => {  
 const photoCard = createPhotoCard(photo);  
 elements.photoGallery.appendChild(photoCard);  
});



### Зміна класів та атрибутів

Управління класами елементів:

// Додавання/видалення класів  
elements.loadMoreBtn.classList.remove('d-none');  
elements.loadMoreBtn.classList.add('d-none');  
// Переключення класу  
btn.classList.toggle('favorited', isFav);

## 

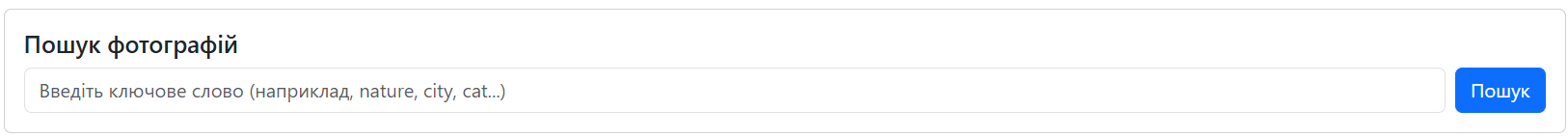
## 3.3 Обробка подій користувача

Реалізовано різні типи обробників подій для інтерактивності.

### Обробка submit форми

Обробка відправки форми пошуку (src/main.js, рядки 255-272):

elements.searchForm.addEventListener('submit', async (e) => {  
 e.preventDefault();  
 const query = elements.searchInput.value.trim();  
 if (!query) {  
 showError('Будь ласка, введіть пошуковий запит');  
 return;  
 }  
 state.currentQuery = query;  
 state.currentPage = 1;  
 state.currentTab = 'all';  
 elements.allTab.classList.add('active');  
 elements.favoritesTab.classList.remove('active');  
 const photos = await fetchPhotos(query, 1);  
 state.photos = photos;  
 displayPhotos(photos);  
});



### Делегування подій

Обробка кліків на галереї з делегуванням (src/main.js, рядки 279-303):

elements.photoGallery.addEventListener('click', (e) => {  
 // Клік на кнопку улюбленого  
 if (e.target.closest('.favorite-btn')) {  
 const btn = e.target.closest('.favorite-btn');  
 const photoId = btn.dataset.photoId;  
 let photo = state.photos.find(p => p.id === photoId);  
 if (!photo) {  
 const favorites = getFavorites();  
 photo = favorites.find(p => p.id === photoId);  
 }  
 if (photo) {  
 toggleFavorite(photo);  
 const isFav = isFavorite(photoId);  
 btn.textContent = isFav ? '❤️' : '🤍';  
 btn.classList.toggle('favorited', isFav);  
 }  
 }  
 // Клік на картку фото  
 else if (e.target.closest('.photo-card')) {

const card = e.target.closest('.photo-card');  
 const photoId = card.dataset.photoId;  
 showPhotoModal(photoId);  
 }  
});

### Обробка прокрутки

Нескінченний скрол (src/main.js, рядки 334-348):

let scrollTimeout;  
function handleScroll() {  
 clearTimeout(scrollTimeout);  
 scrollTimeout = setTimeout(async () => {  
 if (isBottomReached() &&  
 state.currentTab === 'all' &&  
 !state.isLoading) {  
 await loadMorePhotos();  
 }  
 }, 200);  
}  
window.addEventListener('scroll', handleScroll);



## 3.4 Робота з масивами та об’єктами

Активно використовуються методи масивів ES6+ для обробки даних.

### Метод forEach

Ітерація по масиву фотографій (src/main.js, рядки 192-194):

photos.forEach(photo => {  
 const photoCard = createPhotoCard(photo);  
 elements.photoGallery.appendChild(photoCard);  
});

### Метод find

Пошук фотографії за ID (src/main.js, рядки 210-215):

function showPhotoModal(photoId) {  
 let photo = state.photos.find(p => p.id === photoId);  
  
 if (!photo) {  
 const favorites = getFavorites();  
 photo = favorites.find(p => p.id === photoId);  
 }  
 // ...  
}

### Метод some

Перевірка наявності в улюблених (src/main.js, рядки 64-67):

function isFavorite(photoId) {  
 const favorites = getFavorites();  
 return favorites.some(fav => fav.id === photoId);  
}

### Метод findIndex

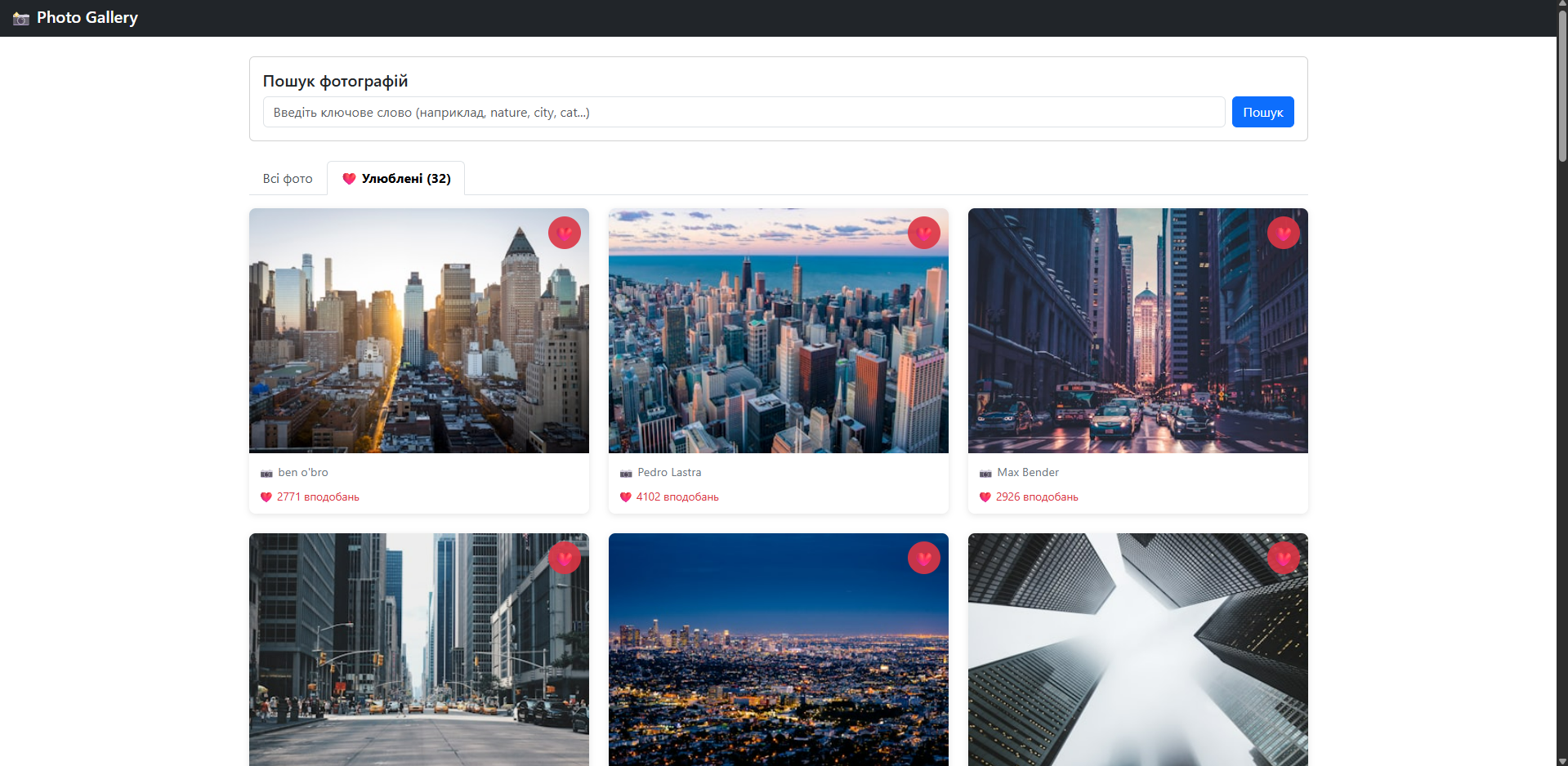
Знаходження індексу для видалення (src/main.js, рядки 70-82):

function toggleFavorite(photo) {  
 const favorites = getFavorites();  
 const index = favorites.findIndex(fav => fav.id === photo.id);  
  
 if (index > -1) {  
 favorites.splice(index, 1); // Видалити  
 } else {  
 favorites.push(photo); // Додати  
 }  
 saveFavorites(favorites);  
}

### Spread operator

Об’єднання масивів (src/main.js, рядок 320):

state.photos = [...state.photos, ...photos];



## 3.5 Розробка HTML-форм та валідація

Створено форму пошуку з валідацією даних.

### HTML форма

Форма пошуку фотографій (index.html, рядки 25-36):

<form id="searchForm" class="d-flex gap-2">  
 <input  
 type="text"  
 id="searchInput"  
 class="form-control"  
 placeholder="Введіть ключове слово (наприклад, nature, city, cat...)"  
 required  
 />  
 <button type="submit" class="btn btn-primary">  
 Пошук  
 </button>  
</form>

### HTML5 валідація

Використано атрибут required для валідації на рівні браузера.

### JavaScript валідація

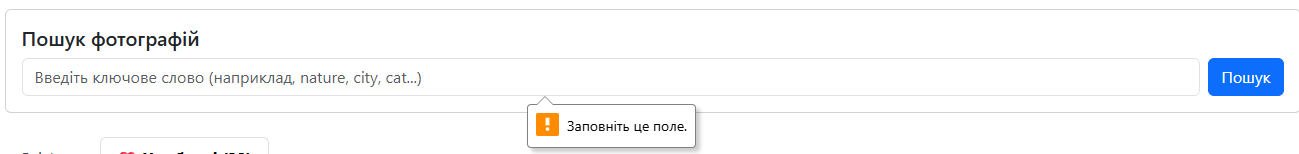
Додаткова валідація в коді (src/main.js, рядки 256-260):

const query = elements.searchInput.value.trim();  
if (!query) {  
 showError('Будь ласка, введіть пошуковий запит');  
 return;  
}

### Зчитування даних з форми

Отримання значення з input:

const query = elements.searchInput.value.trim();



## 

## 3.6 Використання Bootstrap та Axios

Інтеграція сучасних бібліотек для UI та HTTP запитів.

### Підключення Bootstrap

Імпорт Bootstrap в проєкт (src/main.js, рядки 2-5):

import './style.css'  
import 'bootstrap/dist/css/bootstrap.min.css'  
import \* as bootstrap from 'bootstrap'  
import axios from 'axios'

### Bootstrap компоненти

#### Navbar (Навігаційна панель)

<nav class="navbar navbar-dark bg-dark">  
 <div class="container-fluid">  
 <span class="navbar-brand mb-0 h1">📸 Photo Gallery</span>  
 </div>  
</nav>

#### Card (Картка)

<div class="card">  
 <div class="card-body">  
 <h5 class="card-title">Пошук фотографій</h5>  
 <!-- Форма -->  
 </div>  
</div>

Modal (Модальне вікно)

Ініціалізація модального вікна (src/main.js, рядок 32):

photoModal: new bootstrap.Modal(document.getElementById('photoModal'))

Відкриття модального вікна:

elements.photoModal.show();

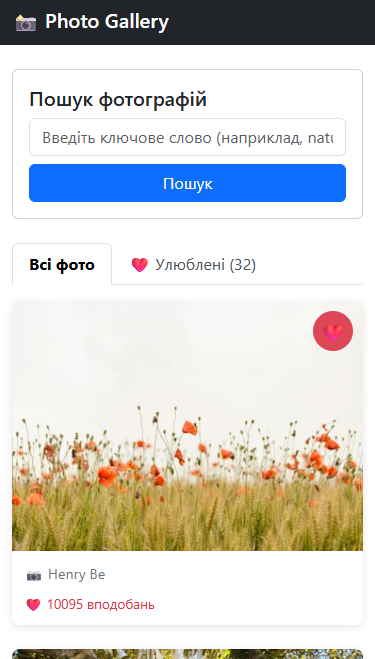
#### Tabs (Вкладки)

<ul class="nav nav-tabs mb-3">  
 <li class="nav-item">  
 <button class="nav-link active" data-tab="all">Всі фото</button>  
 </li>  
 <li class="nav-item">  
 <button class="nav-link" data-tab="favorites">  
 ❤️ Улюблені (<span id="favoritesCount">0</span>)  
 </button>  
 </li>  
</ul>

### Axios для HTTP запитів

GET запит до API (src/main.js, рядки 106-112):

const response = await axios.get(`${UNSPLASH\_API\_URL}/search/photos`, {  
 params: {  
 query: query,  
 page: page,  
 per\_page: 12,  
 client\_id: UNSPLASH\_ACCESS\_KEY  
 }  
});  
return response.data.results;



## 3.7 Взаємодія з зовнішніми API

Реалізовано повноцінну інтеграцію з Unsplash API.

### Конфігурація API

Налаштування доступу (src/main.js, рядки 7-9):

const UNSPLASH\_ACCESS\_KEY = 'EyuRTN2cUCiVfSzRT9Us4oc-kHgYsSuqrFrzqvNDumk';  
const UNSPLASH\_API\_URL = 'https://api.unsplash.com';

### Асинхронна функція запиту

Функція завантаження фотографій (src/main.js, рядки 100-118):

async function fetchPhotos(query = 'nature', page = 1) {  
 try {  
 state.isLoading = true;  
 showLoading(true);  
 hideError();  
 const response = await axios.get(`${UNSPLASH\_API\_URL}/search/photos`, {  
 params: {  
 query: query,  
 page: page,  
 per\_page: 12,  
 client\_id: UNSPLASH\_ACCESS\_KEY  
 }  
 });  
 state.isLoading = false;  
 showLoading(false);  
 return response.data.results;  
 } catch (error) {  
 // Обробка помилок  
 }  
}

Структура відповіді JSON

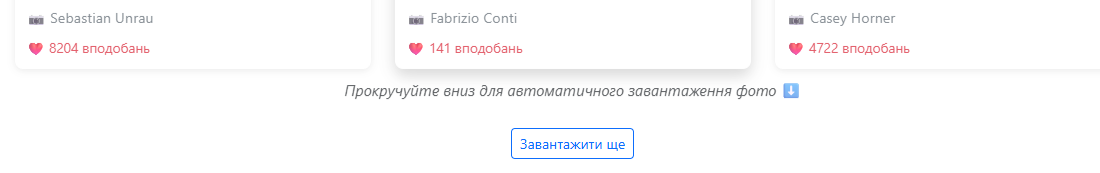
Отримані дані з API:

{  
 "results": [  
 {  
 "id": "photo-id",  
 "urls": {  
 "small": "https://...",  
 "regular": "https://..."  
 },  
 "user": {  
 "name": "Photographer Name",  
 "username": "username"  
 },  
 "likes": 123,  
 "alt\_description": "Description",  
 "description": "Full description"  
 }  
 ]  
}

### Обробка JSON відповіді

Використання даних:

const photos = response.data.results;  
photos.forEach(photo => {  
 console.log(photo.user.name); // Ім'я автора  
 console.log(photo.urls.small); // URL маленького зображення  
 console.log(photo.likes); // Кількість вподобань  
});



## 

## 3.8 Обробка помилок у JavaScript

Реалізовано детальну обробку різних типів помилок.

### Try-catch блоки

Обробка помилок API (src/main.js, рядки 119-142):

} catch (error) {  
 state.isLoading = false;  
 showLoading(false);  
 // Детальна обробка помилок  
 if (error.response) {  
 // Сервер відповів з помилкою  
 if (error.response.status === 401) {  
 showError('Невірний API ключ. Будь ласка, налаштуйте UNSPLASH\_ACCESS\_KEY');  
 } else if (error.response.status === 403) {  
 showError('Перевищено ліміт запитів до API. Спробуйте пізніше.');  
 } else {  
 showError(`Помилка API: ${error.response.status}`);  
 }  
 } else if (error.request) {  
 // Запит відправлений, але відповіді немає  
 showError('Помилка мережі. Перевірте з\'єднання з інтернетом.');  
 } else {  
 // Інша помилка  
 showError('Невідома помилка: ' + error.message);  
 }  
 return [];}

### Async/await синтаксис

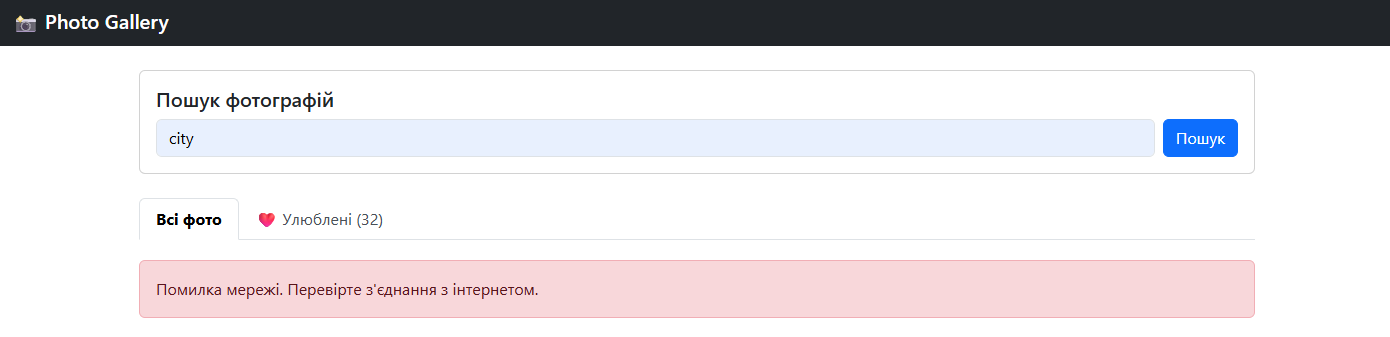
Асинхронні функції:

// Визначення async функції  
async function fetchPhotos(query, page) {  
 const response = await axios.get(url, params);  
 return response.data.results;  
}  
// Виклик async функції  
const photos = await fetchPhotos('nature', 1);

### Обробка помилок localStorage

Захист від помилок (src/main.js, рядки 42-61):

function getFavorites() {  
 try {  
 const favorites = localStorage.getItem('favorites');  
 return favorites ? JSON.parse(favorites) : [];  
 } catch (error) {  
 console.error('Помилка читання з localStorage:', error);  
 return [];  
 }  
}  
function saveFavorites(favorites) {  
 try {  
 localStorage.setItem('favorites', JSON.stringify(favorites));  
 updateFavoritesCount();  
 } catch (error) {  
 console.error('Помилка запису в localStorage:', error);  
 showError('Не вдалося зберегти улюблене фото');  
 }  
}



## 

## 3.9 Пагінація та нескінченний скрол

Реалізовано два способи завантаження додаткових фотографій.

### Функція завантаження

Уніфікована функція (src/main.js, рядки 315-321):

async function loadMorePhotos() {  
 if (state.isLoading || state.currentTab !== 'all') return;  
 state.currentPage++;  
 const photos = await fetchPhotos(state.currentQuery, state.currentPage);  
 state.photos = [...state.photos, ...photos];  
 displayPhotos(photos, true);  
}

### Кнопка “Завантажити ще”

Обробник події кнопки (src/main.js, рядки 310-312):

elements.loadMoreBtn.addEventListener('click', async () => {  
 await loadMorePhotos();  
});

### Infinite Scroll

Перевірка позиції скролу (src/main.js, рядки 323-330):

function isBottomReached() {  
 const scrollTop = window.pageYOffset || document.documentElement.scrollTop;  
 const scrollHeight = document.documentElement.scrollHeight;  
 const clientHeight = document.documentElement.clientHeight;  
 // Завантажуємо за 300px до низу  
 return scrollTop + clientHeight >= scrollHeight - 300;  
}

### Debouncing для оптимізації

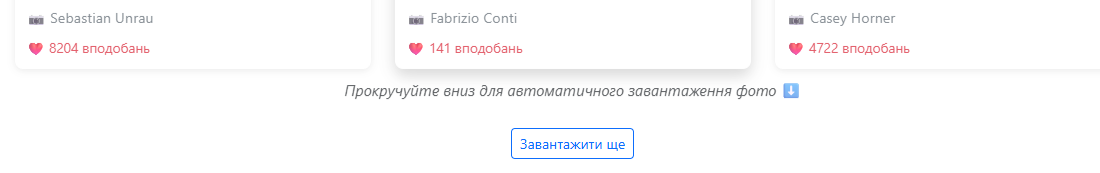
Затримка виконання (src/main.js, рядки 333-346):

let scrollTimeout;  
function handleScroll() {  
 clearTimeout(scrollTimeout);  
 scrollTimeout = setTimeout(async () => {  
 if (isBottomReached() &&  
 state.currentTab === 'all' &&  
 !state.isLoading) {  
 await loadMorePhotos();  
 }  
 }, 200); // Затримка 200мс  
}  
window.addEventListener('scroll', handleScroll);

### Плавна прокрутка

CSS налаштування (src/style.css, рядки 8-10):

html {  
 scroll-behavior: smooth;  
}



## 3.10 Використання localStorage

Реалізовано збереження улюблених фотографій у браузері.

### Збереження даних

Функція збереження (src/main.js, рядки 53-61):

function saveFavorites(favorites) {  
 try {  
 localStorage.setItem('favorites', JSON.stringify(favorites));  
 updateFavoritesCount();  
 } catch (error) {  
 console.error('Помилка запису в localStorage:', error);  
 showError('Не вдалося зберегти улюблене фото');  
 }  
}

### Читання даних

Функція отримання (src/main.js, рядки 42-51):

function getFavorites() {  
 try {  
 const favorites = localStorage.getItem('favorites');  
 return favorites ? JSON.parse(favorites) : [];  
 } catch (error) {  
 console.error('Помилка читання з localStorage:', error);  
 return [];  
 }  
}

### JSON.stringify та JSON.parse

Перетворення об’єктів:

// Збереження масиву об'єктів  
const favorites = [  
 { id: '1', urls: {...}, user: {...} },  
 { id: '2', urls: {...}, user: {...} }  
];  
localStorage.setItem('favorites', JSON.stringify(favorites));  
// Читання та парсинг  
const stored = localStorage.getItem('favorites');  
const favorites = JSON.parse(stored);

### Додавання та видалення

Функція toggleFavorite (src/main.js, рядки 70-88):

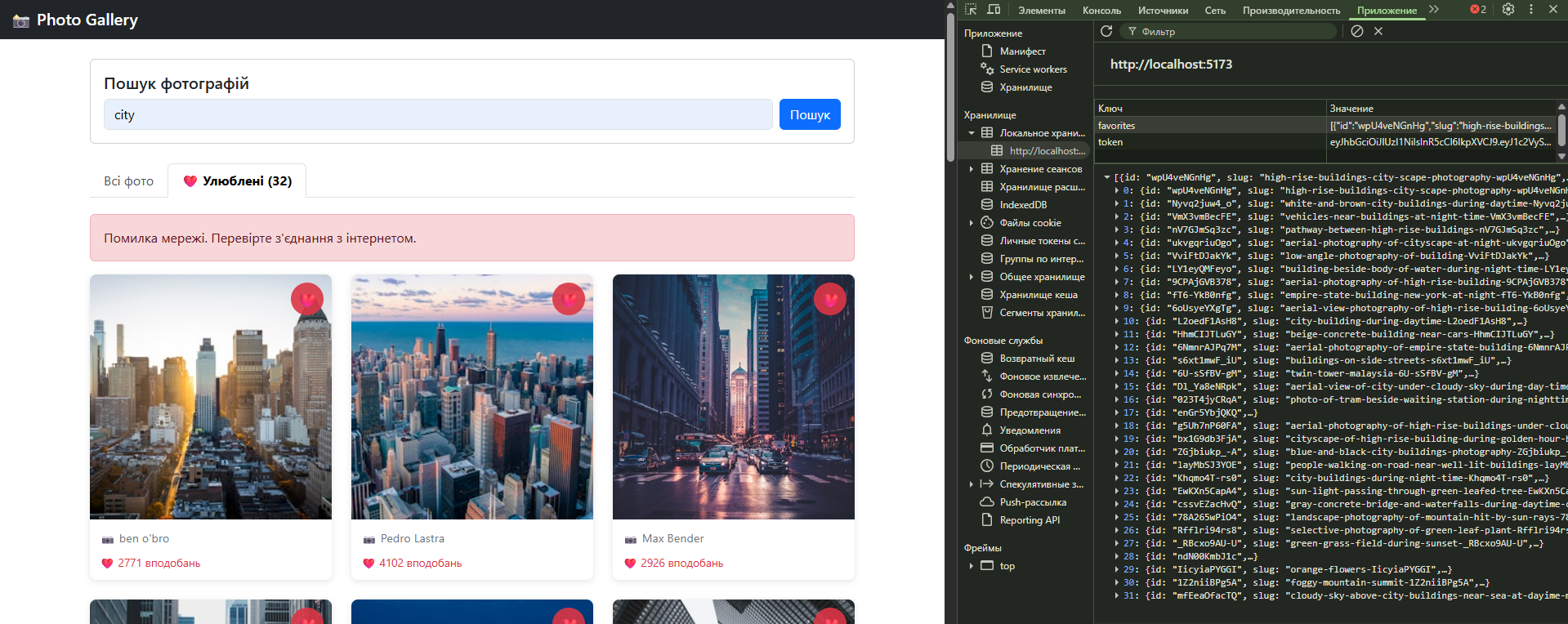
function toggleFavorite(photo) {  
 const favorites = getFavorites();  
 const index = favorites.findIndex(fav => fav.id === photo.id);

if (index > -1) {  
 // Видалити з улюблених  
 favorites.splice(index, 1);  
 } else {  
 // Додати до улюблених  
 favorites.push(photo);  
 }  
 saveFavorites(favorites);  
 // Оновити відображення  
 if (state.currentTab === 'favorites') {  
 displayPhotos(favorites);  
 }  
}

### Оновлення лічильника

Функція підрахунку (src/main.js, рядки 91-94):

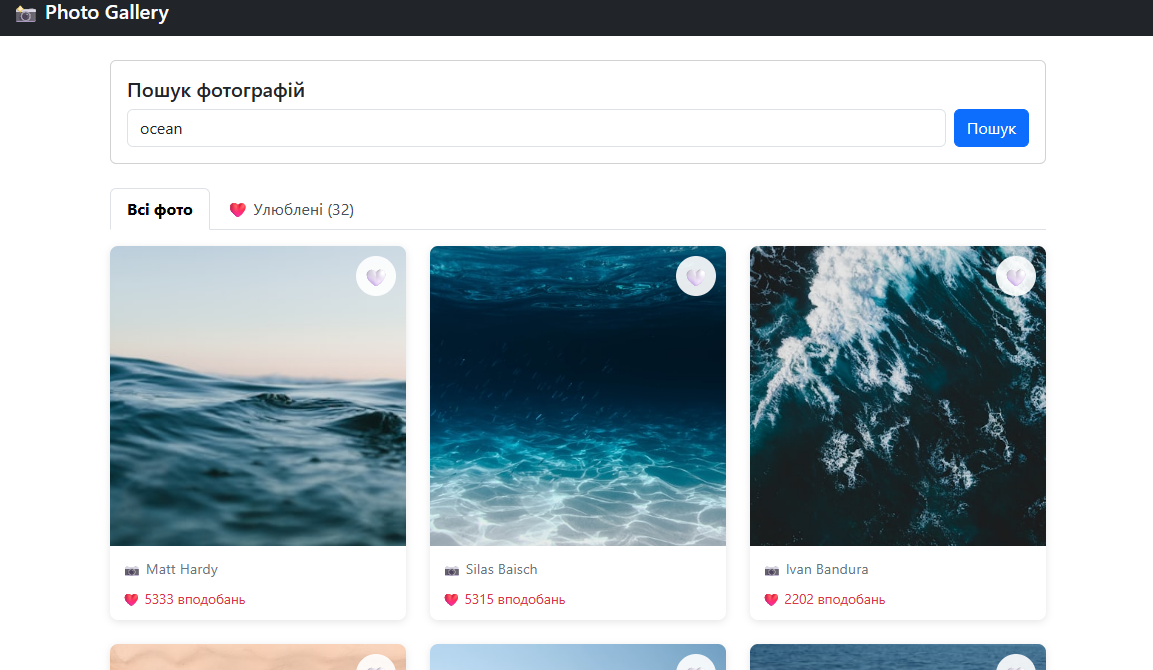
function updateFavoritesCount() {  
 const favorites = getFavorites();  
 elements.favoritesCount.textContent = favorites.length;  
}



**4. ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ**

Користувач може шукати фотографії за ключовими словами:

1. Введення слова в поле пошуку (англійською мовою).
2. Натискання кнопки “Пошук” або Enter.
3. Валідація введення (перевірка на порожнє значення).
4. Запит до Unsplash API з введеним запитом.
5. Відображення результатів у галереї.

Приклади запитів: nature, city, cat, ocean, mountain, sunset.

Перегляд у повному розмірі

При кліку на фотографію відкривається модальне вікно:

1. Велике зображення у високій якості.
2. Опис фотографії (якщо є).
3. Ім’я автора та username.
4. Кількість вподобань.
5. Кнопка “Відкрити оригінал” - посилання на Unsplash.

[СКРІН 14: Детальний вигляд модального вікна]

(Має бути зображення модального вікна з усіма елементами: - великим зображенням - описом під фото - “Автор: John Doe (@johndoe)” - двома кнопками внизу)

## 4.3 Система улюблених

Можливість зберігати улюблені фотографії:

1. Кнопка серця (🤍/❤️) на кожній картці.
2. Клік додає/видаляє фото з улюблених.
3. Дані зберігаються в localStorage.
4. Лічильник улюблених у вкладці.
5. Окрема вкладка “Улюблені” для перегляду.

Збереження після перезавантаження: Улюблені фото залишаються після закриття браузера.

[СКРІН 15: Додавання фото до улюблених]

(Має бути зображення процесу: - картка з білим серцем 🤍 - після кліку - червоне серце ❤️ - оновлений лічильник “Улюблені (1)” → “Улюблені (2)”)

## 4.4 Нескінченний скрол

Автоматичне завантаження фотографій при прокручуванні:

1. Користувач прокручує сторінку вниз.
2. При досягненні позиції за 300px до кінця - автоматичний запит.
3. Завантаження наступної сторінки результатів (12 фото).
4. Додавання нових карток до галереї.
5. Можливість продовжувати скрол.

Оптимізація: Debouncing (200мс) запобігає надмірним запитам.

[СКРІН 16: Процес автозавантаження]

(Має бути зображення: - галереї з багатьма фото - спіннера завантаження внизу - підказки “Прокручуйте вниз…”)

## 4.5 Обробка помилок

Детальні повідомлення для різних ситуацій:

* Порожній запит: “Будь ласка, введіть пошуковий запит”
* Невірний API ключ (401): “Невірний API ключ…”
* Ліміт запитів (403): “Перевищено ліміт запитів…”
* Помилка мережі: “Помилка мережі. Перевірте з’єднання…”
* Помилка localStorage: “Не вдалося зберегти улюблене фото”

[СКРІН 17: Різні типи повідомлень про помилки]

(Має бути колаж з кількома alert повідомленнями: - червоний alert з текстом помилки валідації - червоний alert з текстом помилки API - червоний alert з текстом помилки мережі)

# 5. СТРУКТУРА ПРОЄКТУ

## 5.1 Архітектура файлів

photo-gallery/  
├── public/  
│ └── vite.svg  
├── src/  
│ ├── main.js # Головна логіка (380+ рядків)  
│ └── style.css # Стилі (178 рядків)  
├── .git/ # Git репозиторій  
├── node\_modules/ # Залежності  
├── .gitignore # Git виключення  
├── index.html # HTML структура  
├── package.json # Конфігурація проєкту  
├── package-lock.json # Залежності (lock)  
└── README.md # Документація

## 5.2 Головний файл main.js

Структура коду (380 рядків):

1. Імпорти (рядки 1-5)
   * Bootstrap CSS та JS
   * Axios
   * Власні стилі
2. Конфігурація (рядки 7-18)
   * API ключ Unsplash
   * URL API
   * Стан застосунку (state)
3. DOM елементи (рядки 20-38)
   * Посилання на всі елементи
4. LocalStorage (рядки 40-94)
   * getFavorites()
   * saveFavorites()
   * isFavorite()
   * toggleFavorite()
5. API функції (рядки 96-143)
   * fetchPhotos() з обробкою помилок
6. DOM маніпуляції (рядки 145-247)
   * createPhotoCard()
   * displayPhotos()
   * showPhotoModal()
   * showLoading()
   * showError()
7. Обробники подій (рядки 249-375)
   * searchForm submit
   * photoGallery click
   * loadMoreBtn click
   * Infinite scroll
   * Вкладки tabs
8. Ініціалізація (рядки 377-388)
   * init() функція
   * Запуск застосунку

## 5.3 Файл стилів style.css

Структура CSS (178 рядків):

1. Reset та базові стилі - загальні налаштування
2. Стилі галереї - photo-card з hover ефектами
3. Кнопка улюбленого - favorite-btn
4. Модальне вікно - налаштування розміру
5. Вкладки - стилізація tabs
6. Форма пошуку - responsive layout
7. Анімації - fadeIn, bounce
8. Адаптивність - @media запити

[СКРІН 18: Код проєкту у VS Code]

(Має бути зображення VS Code з: - відкритим файлом main.js - структурою папок зліва - підсвіткою синтаксису JavaScript - номерами рядків)

# 6. ІНСТРУКЦІЯ З ВИКОРИСТАННЯ

## 6.1 Встановлення проєкту

### Крок 1: Клонування репозиторію

git clone https://github.com/AlexBram003/Practika.git  
cd Practika

### Крок 2: Встановлення залежностей

npm install

Буде встановлено: - vite@7.2.4 - axios@1.13.2 - bootstrap@5.3.8

### Крок 3: Налаштування API ключа

1. Зареєструватись на https://unsplash.com/developers
2. Створити новий додаток (New Application)
3. Скопіювати Access Key
4. Відкрити файл src/main.js
5. На рядку 8 замінити:

const UNSPLASH\_ACCESS\_KEY = 'ваш\_access\_key\_тут';

### Крок 4: Запуск dev-сервера

npm run dev

Сервер запуститься на http://localhost:5173

[СКРІН 19: Термінал з запущеним dev-сервером]

(Має бути зображення терміналу з: - командою “npm run dev” - виводом Vite - “Local: http://localhost:5173/” - “ready in XXX ms”)

## 6.2 Використання застосунку

### Пошук фотографій

1. Відкрити http://localhost:5173
2. Ввести ключове слово (англійською)
3. Натиснути “Пошук” або Enter
4. Дочекатись завантаження результатів

### Додавання до улюблених

1. Навести курсор на фото
2. Натиснути іконку серця (🤍)
3. Серце стане червоним (❤️)
4. Лічильник улюблених збільшиться

### Перегляд улюблених

1. Натиснути вкладку “❤️ Улюблені”
2. Переглянути збережені фото
3. Можна видалити, натиснувши ❤️ знову

### Перегляд у повному розмірі

1. Клікнути на будь-яке фото
2. Відкриється модальне вікно
3. Переглянути деталі
4. Натиснути “Відкрити оригінал” для Unsplash
5. Закрити натисканням “Закрити” або ESC

## 6.3 Збірка для production

npm run build

Результат збірки буде в папці dist/

Попередній перегляд:

npm run preview

# 7. ВИСНОВКИ

В результаті виконання навчальної практики з розробки WEB-застосунків було створено повнофункціональний веб-застосунок Photo Gallery з використанням сучасних технологій та інструментів.

## Досягнуті результати

Всі 10 тем практики повністю реалізовані:

1. ✅ Vite, Git, HTML - проєкт ініціалізовано з Vite, створено Git репозиторій, HTML структуру
2. ✅ DOM маніпуляції - querySelector, getElementById, createElement, appendChild
3. ✅ Обробка подій - submit, click, scroll, делегування подій
4. ✅ Масиви та об’єкти - forEach, find, some, findIndex, splice, spread operator
5. ✅ HTML форми - валідація required, обробка submit, зчитування даних
6. ✅ Bootstrap та Axios - Navbar, Card, Modal, Tabs, HTTP запити
7. ✅ Зовнішні API - інтеграція з Unsplash API, обробка JSON
8. ✅ Обробка помилок - try/catch, async/await, різні типи помилок
9. ✅ Пагінація - infinite scroll, debouncing, оптимізація
10. ✅ localStorage - збереження, читання, JSON.parse/stringify

Додаткові досягнення:

* Нескінченний скрол з debouncing оптимізацією
* Адаптивний дизайн для мобільних пристроїв
* Система улюблених з localStorage
* Модальне вікно для детального перегляду
* Детальна обробка помилок з повідомленнями
* Плавні анімації та переходи
* Чистий та документований код
* Повна документація проєкту

## Набуті навички

Технічні навички:

* Робота з Vite для швидкої розробки
* Використання Axios для HTTP запитів
* Інтеграція Bootstrap 5 для UI
* Робота з Unsplash API
* Управління станом застосунку
* Git для контролю версій
* Налагодження в DevTools

Методологічні навички:

* Структурування коду в модулі
* Асинхронне програмування
* Обробка помилок
* Оптимізація продуктивності
* Документування коду
* Написання README

## Практична цінність

Створений застосунок демонструє:

* Реальну інтеграцію з зовнішнім API
* Сучасні підходи до веб-розробки
* Best practices JavaScript ES6+
* UX/UI принципи
* Готовність до використання

Проєкт повністю відповідає методичним рекомендаціям та охоплює всі обов’язкові теми практики з детальними прикладами коду та демонстрацією функціональності.

[СКРІН 20: Фінальний вигляд застосунку]

(Має бути зображення повного екрану з: - навігацією - формою пошуку - заповненою галереєю - всіма елементами інтерфейсу)

# 8. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

## Офіційна документація

1. Unsplash API Documentation https://unsplash.com/documentation Документація API Unsplash для роботи з фотографіями
2. Vite Documentation https://vitejs.dev/guide/ Офіційна документація збирача Vite
3. Bootstrap 5 Documentation https://getbootstrap.com/docs/5.3/ Документація фреймворку Bootstrap 5
4. Axios Documentation https://axios-http.com/docs/intro Документація бібліотеки Axios для HTTP запитів
5. MDN Web Docs - JavaScript https://developer.mozilla.org/en-US/docs/Web/JavaScript Повний довідник JavaScript
6. MDN Web Docs - HTML https://developer.mozilla.org/en-US/docs/Web/HTML Документація HTML
7. MDN Web Docs - CSS https://developer.mozilla.org/en-US/docs/Web/CSS Документація CSS

## Навчальні матеріали

1. JavaScript.info - Сучасний підручник JavaScript https://uk.javascript.info/ Детальний курс JavaScript українською мовою
2. W3Schools - JavaScript Tutorial https://www.w3schools.com/js/ Інтерактивні уроки JavaScript
3. Eloquent JavaScript (4th Edition) - Marijn Haverbeke https://eloquentjavascript.net/ Книга про JavaScript з прикладами

## Методичні рекомендації

1. Навчальна практика з розробки WEB-застосунків Методичні рекомендації щодо виконання практичних завдань Олександрійський політехнічний фаховий коледж, 2025

## Інструменти розробки

1. Git Documentation https://git-scm.com/doc Документація системи контролю версій Git
2. NPM Documentation https://docs.npmjs.com/ Документація менеджера пакетів npm
3. Chrome DevTools https://developer.chrome.com/docs/devtools/ Інструменти розробника Chrome
4. Visual Studio Code https://code.visualstudio.com/docs Документація редактора VS Code