НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №1**

з дисципліни **Програмування інтерфейсів користувача**

*на тему: “Розробка клієнтської частини Web-додатка”*

Виконав:

студент V курсу

групи КВ-32мп

Неживий Богдан Миколайович

Перевірив: Петрашенко А. В.

Київ – 2024

**Загальна інформація**

**Завдання на лабораторну роботу***:* розробити клієнтську частину Web-додатку.

[Посилання на Github-репозиторій](https://github.com/n0tark/user-interfaces);

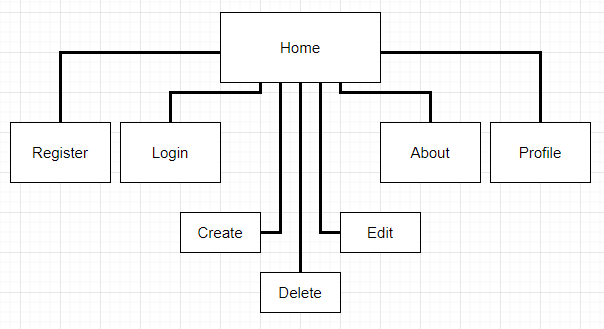
*Посилання студента в* [Telegram](https://t.me/unalive1);

**Назва додатку***:* ContactSphere

**Короткий опис функціональності***:* Цей додаток надає зручний інтерфейс для керування списком контактів, що дозволяє користувачам додавати нові контакти, видаляти та редагувати існуючі записи.

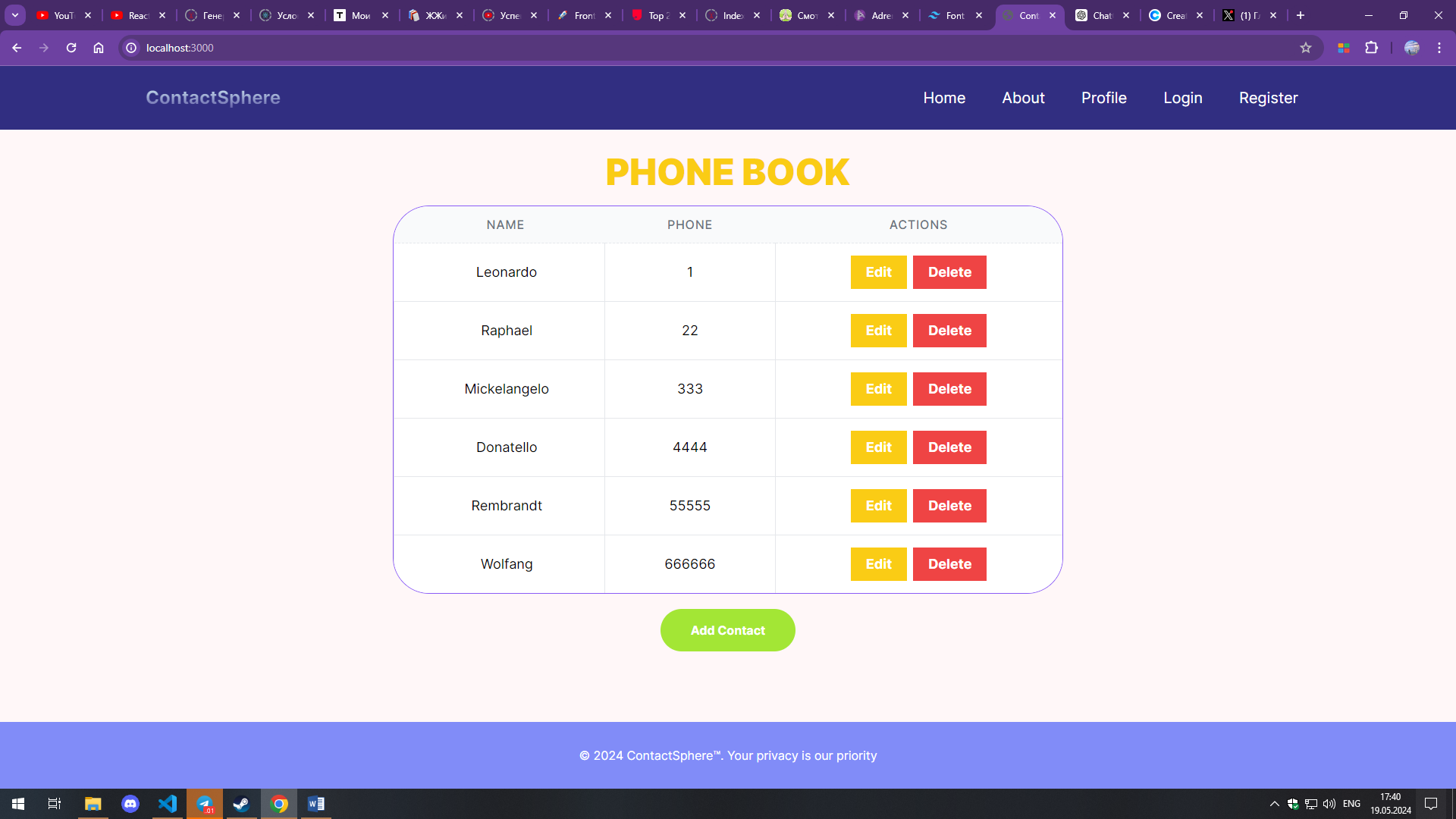
**Технології, використані в розробці***:* Tailwind CSS, JSON Server, Next.ts, Material Icons.

**Карта сайту**

****

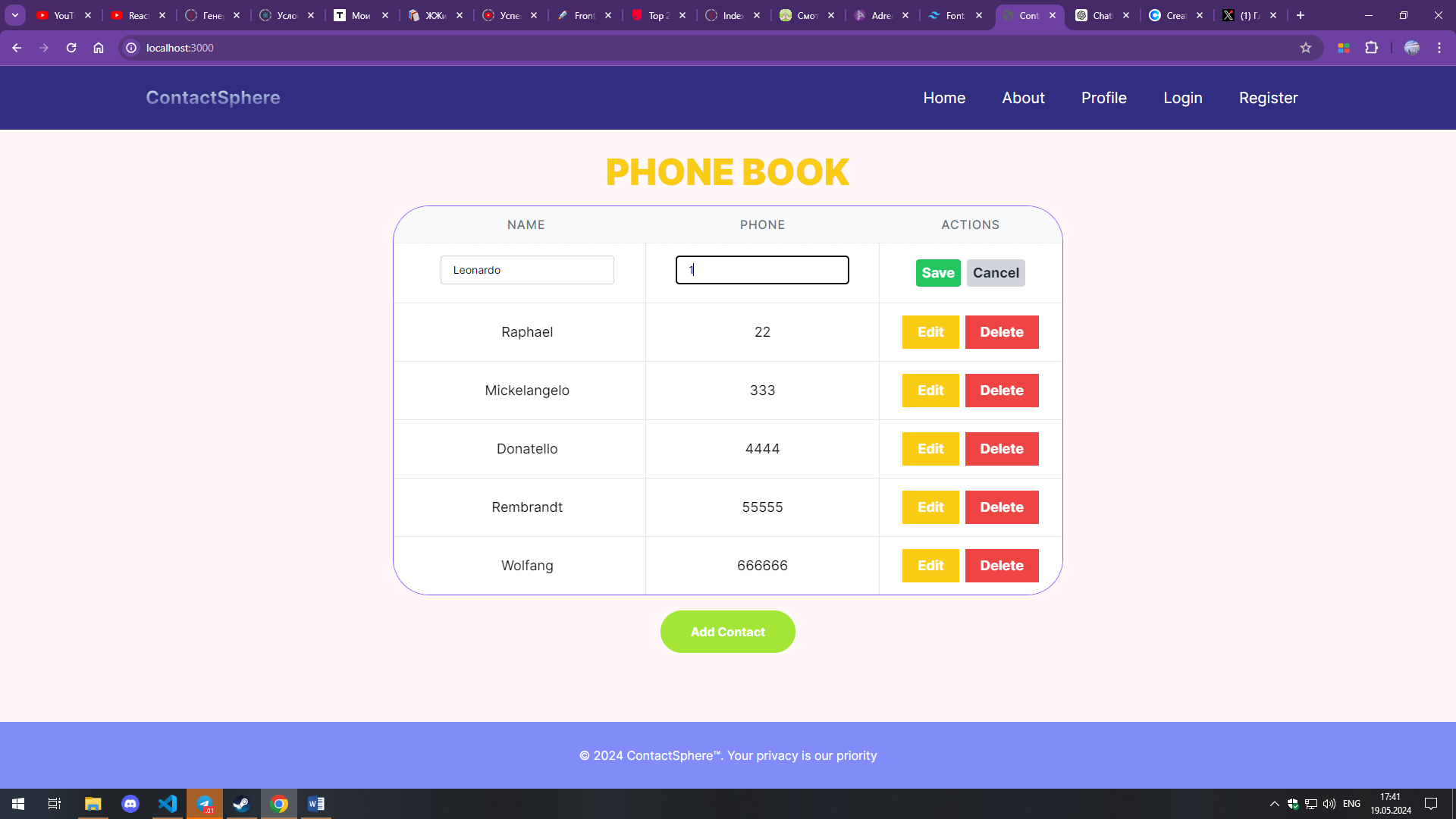
**Скрiншоти всіх сторінок Web-додатку**

**Робоча сторінка**

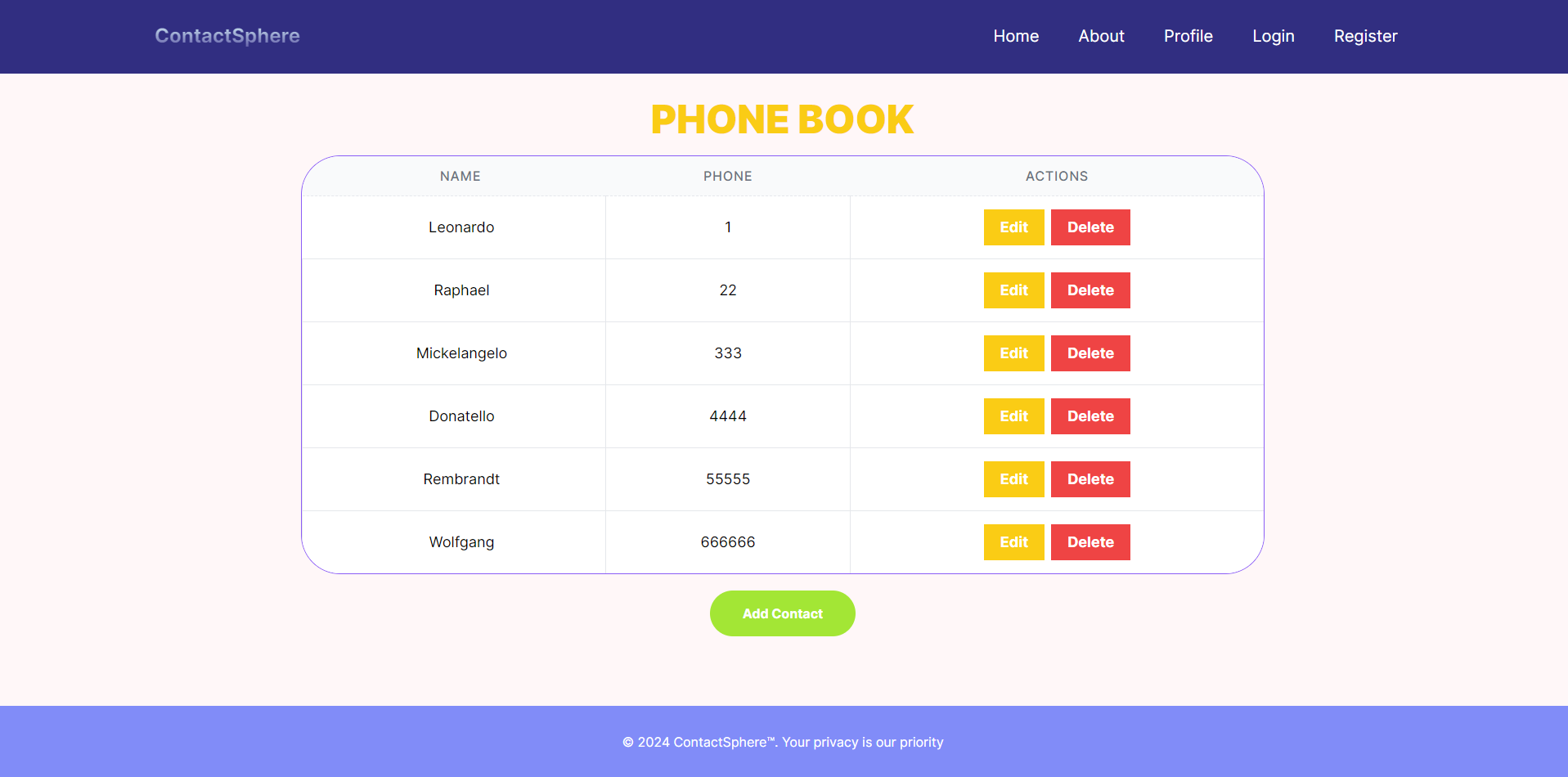


**Демонстрація роботи:**

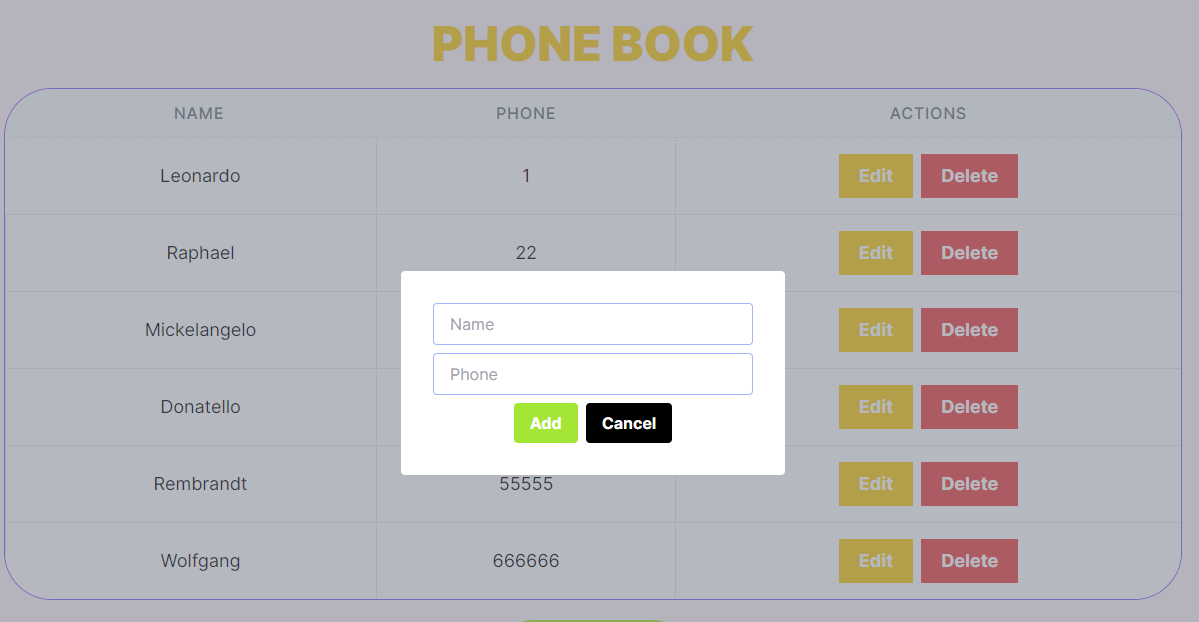
**Редагування**



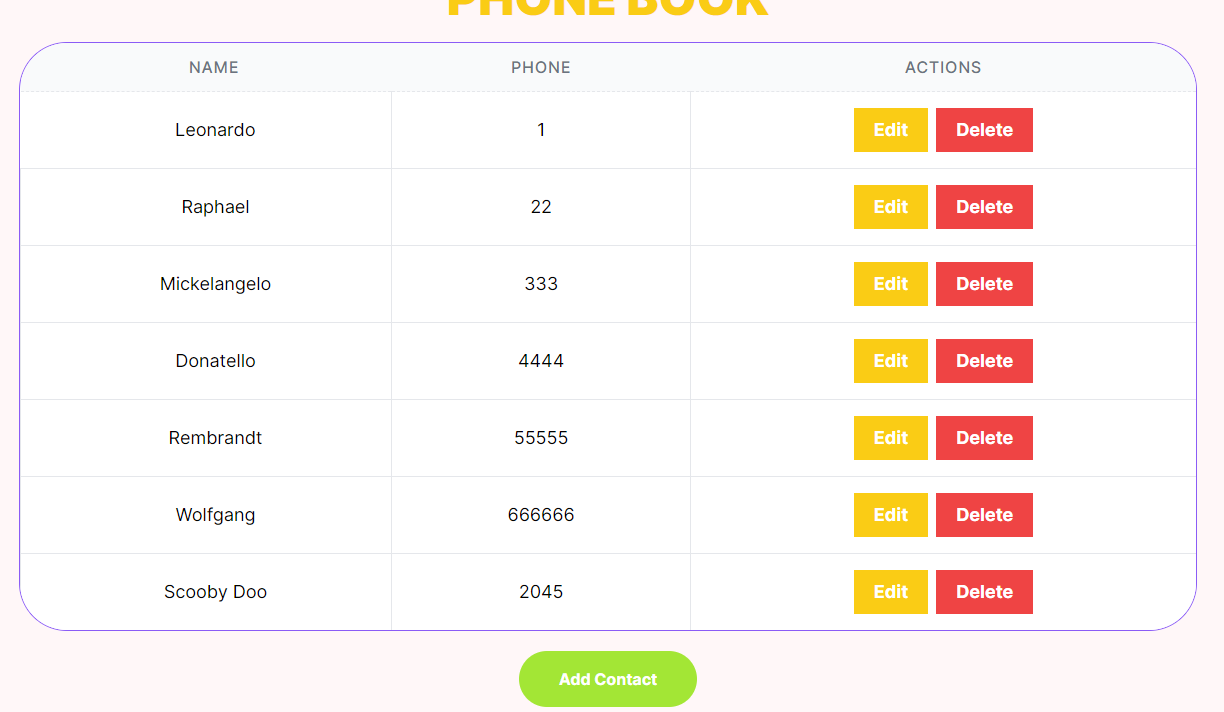
**Відсутність даних**



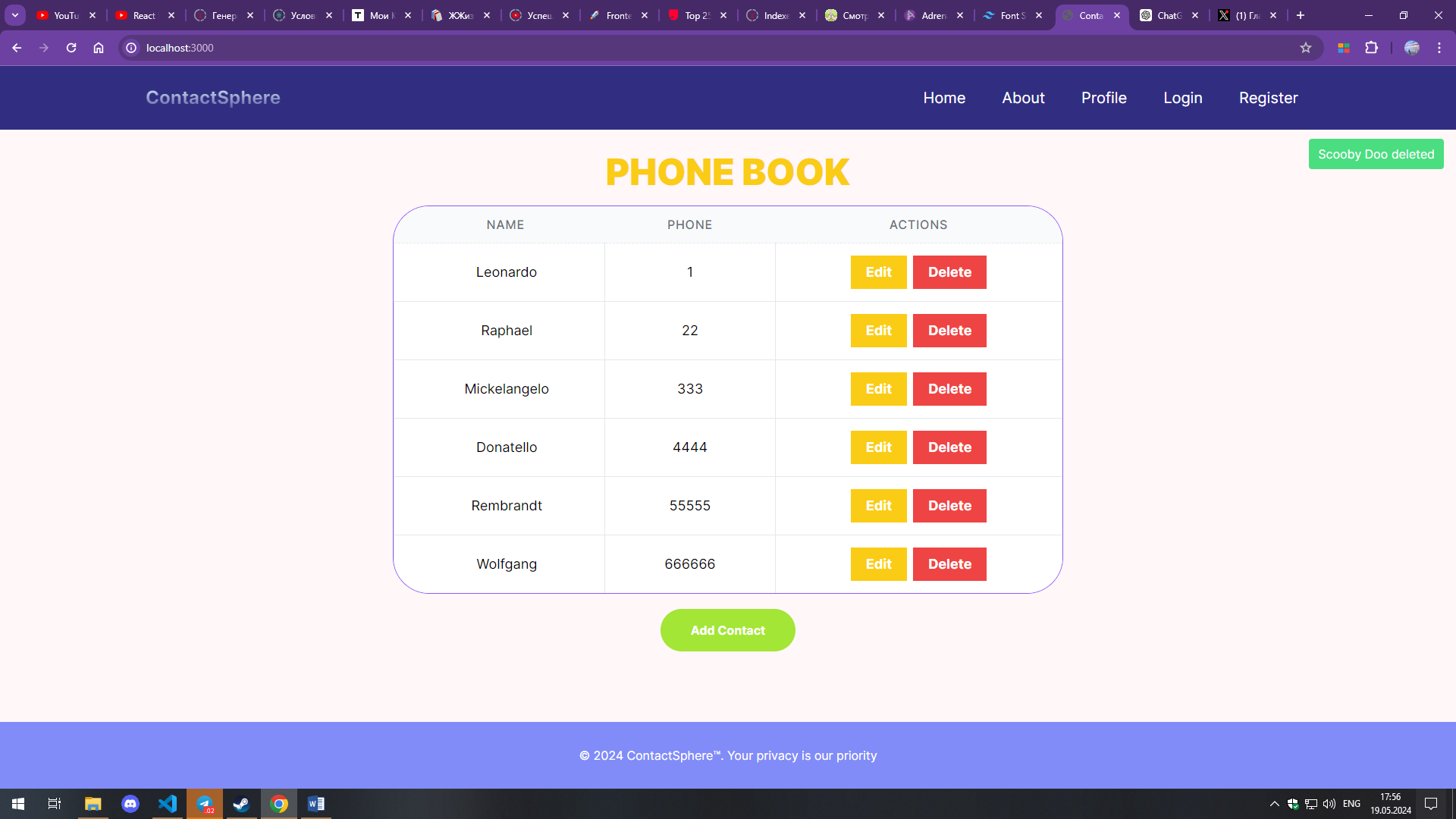
**Додавання контакту**



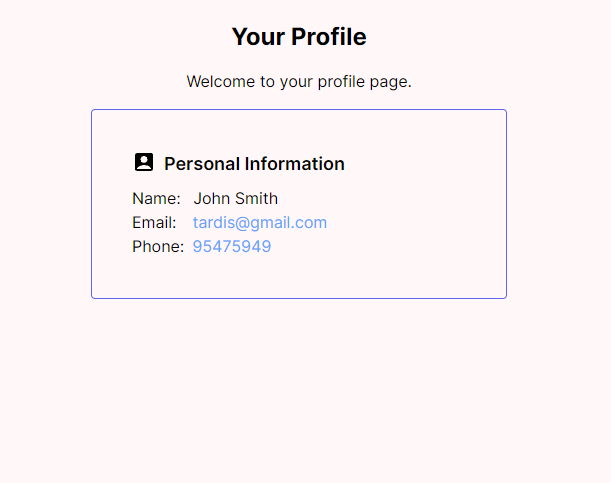
**Результат додавання**



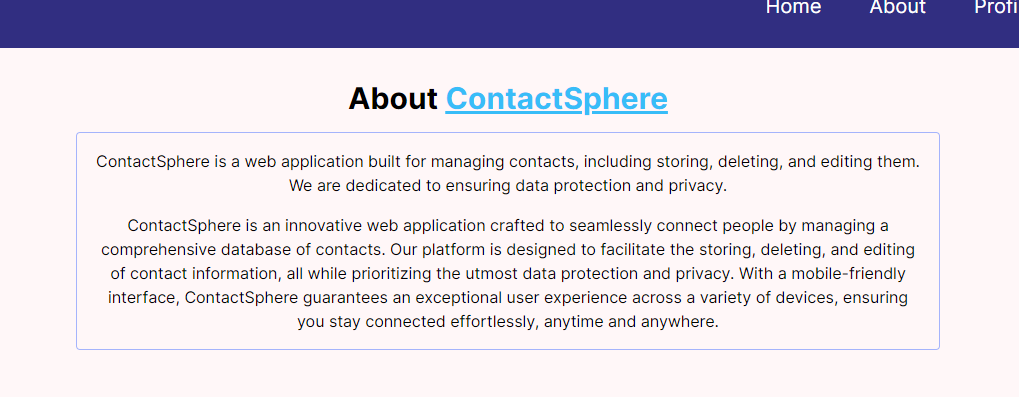
**Видалення останнього контакту**



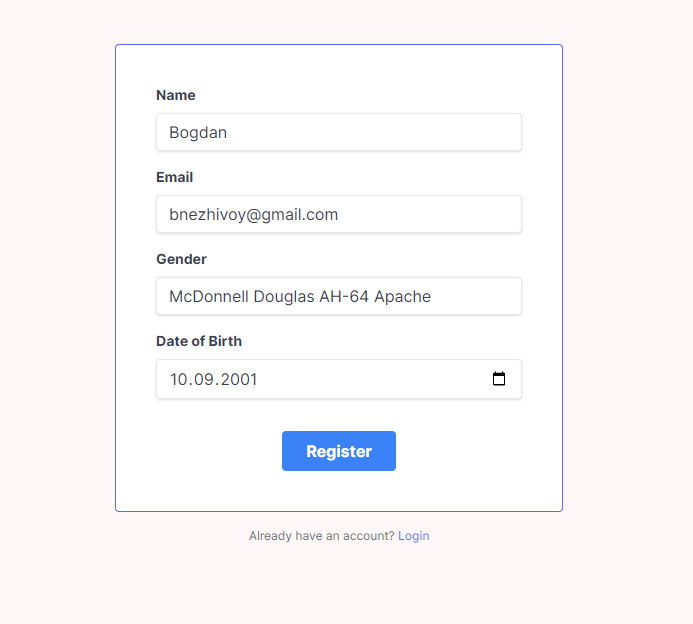
**Профіль**



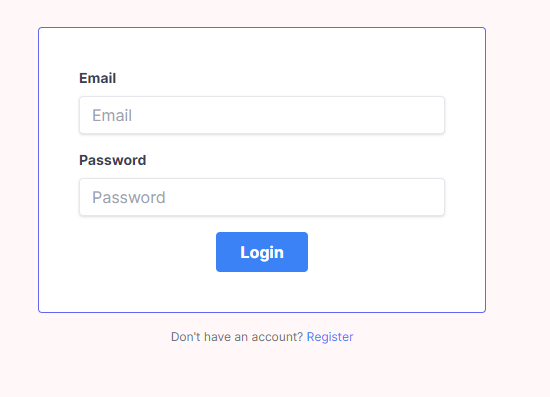
**Про додаток**



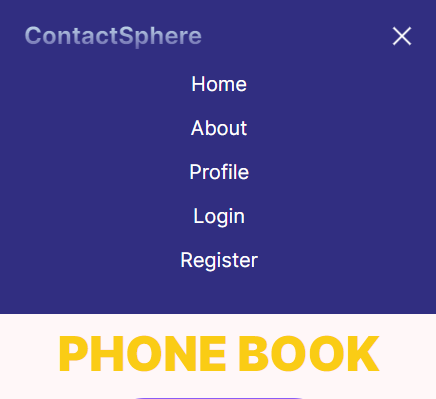
**Сторінка реєстрації**

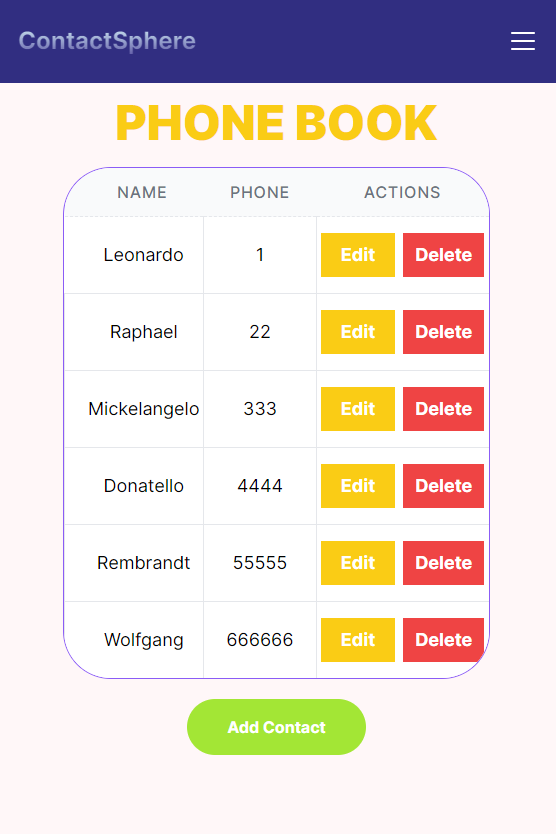


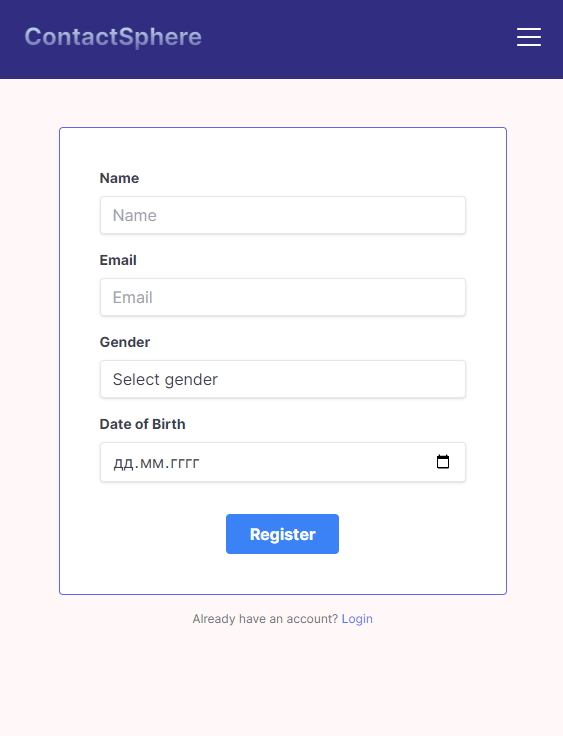
**Сторінка входу**

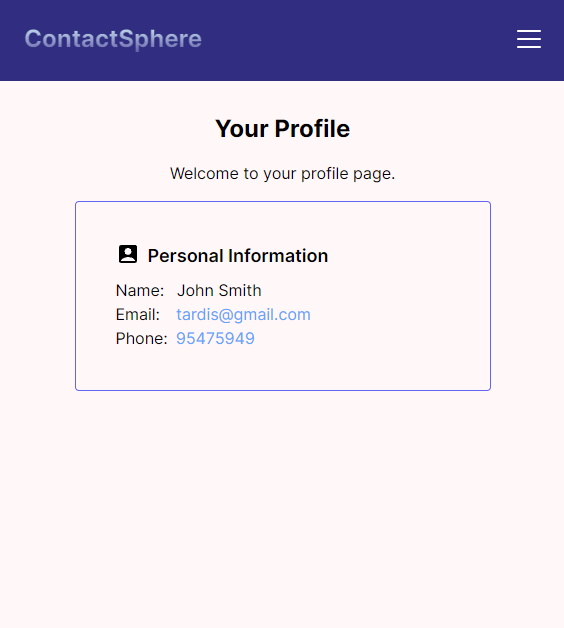


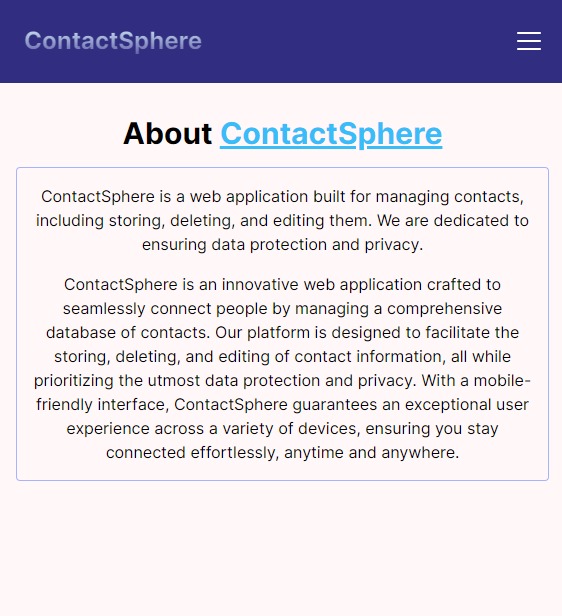
**Скрiншоти всіх сторінок на мобільній версії**











**Фрагменти програмного коду**

**layout.tsx**

import type { Metadata } from 'next';

import { Inter } from 'next/font/google';

import './globals.css';

import Footer from '@/components/common/footer';

import Nav from '@/components/common/nav';

const inter = Inter({ subsets: ['latin'] });

export const metadata: Metadata = {

    title: 'ContactSphere',

    description: 'Nezhyvyi Bogdan KV32mp',

};

export default function RootLayout({

    children,

}: Readonly<{

    children: React.ReactNode;

}>) {

    return (

        <html lang="en">

            <body className={inter.className} style={{ background: '#FFF7F8' }}>

                <Nav />

                {children}

                <Footer />

            </body>

        </html>

    );

}

**Nav.tsx**

'use client';

import React, { useState } from 'react';

import Link from 'next/link';

import '/css/menuButton.css';

const MobileMenuLinks = () => (

    <>

        <Link

            href="/"

            className="block text-xl text-white py-2 px-4 hover:overline"

        >

            Home

        </Link>

        <Link

            href="/about"

            className="block text-xl text-white py-2 px-4 hover:overline"

        >

            About

        </Link>

        <Link

            href="/profile"

            className="block text-xl text-white py-2 px-4 hover:overline"

        >

            Profile

        </Link>

        <Link

            href="/login"

            className="block text-xl text-white py-2 px-4 hover:overline"

        >

            Login

        </Link>

        <Link

            href="/register"

            className="block text-xl text-white py-2 px-4 hover:overline "

        >

            Register

        </Link>

    </>

);

const Nav = () => {

    const [isOpen, setIsOpen] = useState(false);

    return (

        <nav className="bg-indigo-900 p-6">

            <div className="container mx-auto flex justify-between items-center">

                <div className="flex items-center">

                    <div className="font-bold">

                        <Link href="/" className="block max-w-fit">

                            <div

                                style={{

                                    fontSize: '24px',

                                    background:

                                        '-webkit-gradient(linear, left top, left bottom, from(#DDFDFE), to(#52489C))',

                                    WebkitBackgroundClip: 'text',

                                    WebkitTextFillColor: 'transparent',

                                    display: 'inline-block',

                                }}

                            >

                                ContactSphere

                            </div>

                        </Link>

                    </div>

                </div>

                <div className="hidden md:flex space-x-4">

                    <MobileMenuLinks />

                </div>

                <div className="md:hidden">

                    <button

                        className={`text-white focus:outline-none hamburger ${

                            isOpen ? 'open' : ''

                        }`}

                        onClick={() => setIsOpen(!isOpen)}

                    >

                        <span className="bar"></span>

                        <span className="bar"></span>

                        <span className="bar"></span>

                    </button>

                </div>

            </div>

            {isOpen && (

                <div className="md:hidden py-2 text-center">

                    <MobileMenuLinks />

                </div>

            )}

        </nav>

    );

};

export default Nav;

**footer.tsx**

import React from 'react';

const Footer = () => {

    return (

        <footer className="bg-indigo-400 text-white py-8 text-center absolute bottom-0 w-full">

            <div className="container mx-auto">

                <p>

                    &copy; 2024 ContactSphere&#8482;. Your privacy is our

                    priority

                </p>

            </div>

        </footer>

    );

};

export default Footer;

**Contacts.tsx**

import React from 'react';

import ContactList from '@/components/contacts/contactList';

const Contacts: React.FC = () => {

    return (

        <div>

            <ContactList />

        </div>

    );

};

export default Contacts;

**contactList.tsx**

'use client';

import React, { useState, useEffect, useRef } from 'react';

import {

    addContact,

    deleteContact,

    getContacts,

    updateContact,

    resetContacts,

} from '@/components/contacts/contactProp';

export const ContactList: React.FC = () => {

    const [contacts, setContacts] = useState<any[]>([]);

    const [editContact, setEditContact] = useState<any>(null);

    const [addContactModal, setAddContactModal] = useState(false);

    const [newContact, setNewContact] = useState({ name: '', phones: '' });

    const [deleteNotification, setDeleteNotification] = useState(false);

    const [deletedContact, setDeletedContact] = useState<any>(null);

    const prevEditContact = useRef<any>(null);

    useEffect(() => {

        async function fetchData() {

            try {

                const data = await getContacts();

                setContacts(data);

            } catch (error) {

                console.error(error);

            }

        }

        fetchData();

    }, []);

    const handleEdit = async (contact: any) => {

        setEditContact(contact);

        prevEditContact.current = { ...contact };

    };

    const handleSaveEdit = async () => {

        if (editContact) {

            await updateContact(editContact.id, editContact);

            setEditContact(null);

        }

        const updatedContacts = await getContacts();

        setContacts(updatedContacts);

    };

    const handleCancelEdit = () => {

        if (prevEditContact.current) {

            setEditContact(prevEditContact.current);

            prevEditContact.current = null;

        }

    };

    const handleDelete = async (id: string) => {

        const deletedContact = contacts.find((contact) => contact.id === id);

        await deleteContact(id);

        setContacts(contacts.filter((contact) => contact.id !== id));

        setDeletedContact(deletedContact);

        setDeleteNotification(true);

        setTimeout(() => {

            setDeleteNotification(false);

            setDeletedContact(null);

        }, 3000);

    };

    const handleAdd = async () => {

        try {

            await addContact(newContact);

            const updatedContacts = await getContacts();

            setContacts(updatedContacts);

            setNewContact({ name: '', phones: '' });

            setAddContactModal(false);

        } catch (error) {

            console.error('Error adding contact:', error);

        }

    };

    const handleReset = async () => {

        try {

            const newContacts = await resetContacts();

            setContacts(newContacts);

            console.log(newContacts);

        } catch (error) {

            console.error('Error resetting contacts:', error);

        }

    };

    return (

        <div className="container mx-auto p-4 sm:p-8">

            <h1 className="text-5xl font-black uppercase text-yellow-400 text-center mb-5 ">

                Phone Book

            </h1>

            {deleteNotification && deletedContact && (

                <div className="absolute top-24 right-4 bg-green-400 text-white px-3 py-2 rounded">

                    {deletedContact.name} deleted

                </div>

            )}

            {contacts.length === 0 ? (

                <div className="flex justify-center items-center h-64">

                    <h1 className="text-black font-bold text-3xl">

                        Oops! Seems like you have no contacts.

                    </h1>

                </div>

            ) : (

                <>

                    <div

                        className="bg-white border border-violet-500 ms-auto me-auto overflow-hidden"

                        style={{ width: '80%', borderRadius: '3rem' }}

                    >

                        <table className="ms-auto me-auto min-w-full divide-y divide-dashed divide-gray-200 text-lg">

                            <thead className="bg-gray-50 p-5 divide-x text-base">

                                <tr>

                                    <th className="px-6 ps-10 py-3 text-center font-medium text-gray-500 uppercase tracking-wider">

                                        Name

                                    </th>

                                    <th className="px-6 py-3 text-center font-medium text-gray-500 uppercase tracking-wider">

                                        Phone

                                    </th>

                                    <th className="px-6 py-3 text-center font-medium text-gray-500 uppercase tracking-wider">

                                        Actions

                                    </th>

                                </tr>

                            </thead>

                            <tbody className="bg-white min-w-full divide-y divide-x divide-gray-200">

                                {contacts.map((contact) => (

                                    <tr key={contact.id} className="divide-x">

                                        <td className="py-4 ps-5 text-center ">

                                            {editContact?.id === contact.id ? (

                                                <input

                                                    type="text"

                                                    className="text-sm border border-gray-300 rounded px-4 py-2 mb-2"

                                                    value={

                                                        editContact?.name || ''

                                                    }

                                                    onChange={(e) =>

                                                        setEditContact(

                                                            (

                                                                prevEditContact: any

                                                            ) => ({

                                                                ...prevEditContact,

                                                                name: e.target

                                                                    .value,

                                                            })

                                                        )

                                                    }

                                                />

                                            ) : (

                                                contact.name

                                            )}

                                        </td>

                                        <td className="py-4 text-center">

                                            {editContact?.id === contact.id ? (

                                                <input

                                                    type="text"

                                                    className="text-sm border border-gray-300 rounded px-4 py-2 mb-2"

                                                    value={

                                                        editContact?.phones ||

                                                        ''

                                                    }

                                                    onChange={(e) =>

                                                        setEditContact(

                                                            (

                                                                prevEditContact: any

                                                            ) => ({

                                                                ...prevEditContact,

                                                                phones: e.target

                                                                    .value,

                                                            })

                                                        )

                                                    }

                                                />

                                            ) : (

                                                contact.phones

                                            )}

                                        </td>

                                        <td className="py-4 text-center">

                                            {editContact?.id === contact.id ? (

                                                <>

                                                    <button

                                                        className="bg-green-500 hover:bg-green-700 transition-color duration-500 text-white font-bold py-1 px-2 rounded"

                                                        onClick={handleSaveEdit}

                                                    >

                                                        Save

                                                    </button>

                                                    <button

                                                        className="bg-gray-300 hover:bg-gray-400 transition-color duration-500 text-gray-800 font-bold py-1 px-2 rounded ml-2"

                                                        onClick={

                                                            handleCancelEdit

                                                        }

                                                    >

                                                        Cancel

                                                    </button>

                                                </>

                                            ) : (

                                                <>

                                                    <button

                                                        className="bg-yellow-400 hover:bg-yellow-500 transition-color duration-500 text-white font-bold py-2 px-5 "

                                                        onClick={() =>

                                                            handleEdit(contact)

                                                        }

                                                    >

                                                        Edit

                                                    </button>

                                                    <button

                                                        className="bg-red-500 hover:bg-red-700 transition-color duration-500 text-white font-bold py-2 px-3 ml-2 text-center"

                                                        onClick={() =>

                                                            handleDelete(

                                                                contact.id

                                                            )

                                                        }

                                                    >

                                                        Delete

                                                    </button>

                                                </>

                                            )}

                                        </td>

                                    </tr>

                                ))}

                            </tbody>

                        </table>

                    </div>

                    <div className="mb-4">

                        <button

                            className="block mt-5 ms-auto me-auto bg-lime-400 hover:bg-lime-600 transition-color duration-500 text-white font-bold py-4 px-10 rounded-full"

                            onClick={() => setAddContactModal(true)}

                        >

                            Add Contact

                        </button>

                    </div>

                    {/\* <div className="mt-5">

                        <button

                            className="block mt-5 ms-auto me-auto bg-rose-600 hover:bg-rose-800 transition-color duration-500 text-amber-400 font-bold py-4 px-10 rounded-full"

                            onClick={handleReset}

                        >

                            Reset Contacts

                        </button>

                    </div> \*/}

                </>

            )}

            {addContactModal && (

                <div className="fixed top-0 left-0 right-0 bottom-0 flex items-center justify-center bg-gray-500 bg-opacity-50">

                    <div className="bg-white p-4 rounded sm:w-96 sm:p-8 ">

                        <input

                            className="text-base w-full border border-indigo-300 rounded px-4 py-2 mb-2 outline-none"

                            type="text"

                            placeholder="Name"

                            value={newContact.name}

                            onChange={(e) =>

                                setNewContact({

                                    ...newContact,

                                    name: e.target.value,

                                })

                            }

                        />

                        <input

                            className="text-base w-full border border-indigo-300 rounded px-4 py-2 mb-2 outline-none"

                            type="text"

                            placeholder="Phone"

                            value={newContact.phones}

                            onChange={(e) =>

                                setNewContact({

                                    ...newContact,

                                    phones: e.target.value,

                                })

                            }

                        />

                        <div className="flex justify-center">

                            <button

                                className="bg-lime-400 hover:bg-lime-600 transition-color duration-500 text-white font-bold py-2 px-4 rounded mr-2"

                                onClick={handleAdd}

                            >

                                Add

                            </button>

                            <button

                                className="bg-black hover:bg-slate-800 transition-color duration-500 text-white font-bold py-2 px-4 rounded"

                                onClick={() => setAddContactModal(false)}

                            >

                                Cancel

                            </button>

                        </div>

                    </div>

                </div>

            )}

        </div>

    );

};

export default ContactList;

**contactProp.ts**

export async function getContacts() {

    const res = await fetch('http://localhost:4000/contacts');

    const data = await res.json();

    return data.contacts;

}

export async function updateContact(id: string, data: any) {

    await fetch(`http://localhost:4000/contacts/${id}`, {

        method: 'PUT',

        headers: {

            'Content-Type': 'application/json',

        },

        body: JSON.stringify(data),

    });

}

export async function deleteContact(id: string) {

    await fetch(`http://localhost:4000/contacts/${id}`, {

        method: 'DELETE',

    });

}

export async function addContact(data: any) {

    try {

        const res = await fetch('http://localhost:4000/contacts', {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json',

            },

            body: JSON.stringify(data),

        });

        if (res.ok) {

            await getContacts();

            const updatedContacts = await res.json();

            localStorage.setItem('contacts', JSON.stringify(updatedContacts));

        } else {

            console.error('Failed to add contact:', res.status);

        }

    } catch (error) {

        console.error('Error adding contact:', error);

    }

}

export async function resetContacts() {

    const res = await fetch('http://localhost:4000/contacts/reset', {

        method: 'POST',

    });

    if (!res.ok) {

        throw new Error('Failed to reset contacts');

    }

    const data = await res.json();

    return data.contacts;

}

**Register.tsx**

'use client';

import React, { useState } from 'react';

import Link from 'next/link';

const Register: React.FC = () => {

    const [name, setName] = useState('');

    const [email, setEmail] = useState('');

    const [gender, setGender] = useState('');

    const [birthdate, setBirthdate] = useState('');

    const handleSubmit = (e: React.FormEvent<HTMLFormElement>) => {

        e.preventDefault();

    };

    return (

        <div className="flex justify-center mt-12">

            <div className="w-full max-w-md">

                <form

                    onSubmit={handleSubmit}

                    className="bg-white border border-indigo-500 rounded p-10 mb-4"

                >

                    <div className="mb-4">

                        <label

                            className="block text-gray-700 text-sm font-bold mb-2"

                            htmlFor="name"

                        >

                            Name

                        </label>

                        <input

                            id="name"

                            type="text"

                            placeholder="Name"

                            value={name}

                            onChange={(e) => setName(e.target.value)}

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

                        />

                    </div>

                    <div className="mb-4">

                        <label

                            className="block text-gray-700 text-sm font-bold mb-2"

                            htmlFor="email"

                        >

                            Email

                        </label>

                        <input

                            id="email"

                            type="email"

                            placeholder="Email"

                            value={email}

                            onChange={(e) => setEmail(e.target.value)}

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

                        />

                    </div>

                    <div className="mb-4">

                        <label

                            className="block text-gray-700 text-sm font-bold mb-2"

                            htmlFor="gender"

                        >

                            Gender

                        </label>

                        <select

                            id="gender"

                            value={gender}

                            onChange={(e) => setGender(e.target.value)}

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

                        >

                            <option value="">Select gender</option>

                            <option value="male">Male</option>

                            <option value="female">Female</option>

                            <option value="other">

                                McDonnell Douglas AH-64 Apache

                            </option>

                        </select>

                    </div>

                    <div className="mb-8">

                        <label

                            className="block text-gray-700 text-sm font-bold mb-2"

                            htmlFor="birthdate"

                        >

                            Date of Birth

                        </label>

                        <input

                            id="birthdate"

                            type="date"

                            placeholder="Date of Birth"

                            value={birthdate}

                            onChange={(e) => setBirthdate(e.target.value)}

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

                        />

                    </div>

                    <div className="flex items-center justify-between">

                        <button

                            type="submit"

                            className="bg-blue-500 hover:bg-green-700 transition-color duration-500 text-white font-bold py-2 px-6 ms-auto me-auto rounded"

                        >

                            Register

                        </button>

                    </div>

                </form>

                <p className="text-center text-gray-500 text-xs">

                    Already have an account?{' '}

                    <Link

                        href="/login"

                        className="text-blue-500 hover:text-blue-700"

                    >

                        Login

                    </Link>

                </p>

            </div>

        </div>

    );

};

export default Register;

**Profile.tsx**

import React from 'react';

import AccountBoxIcon from '@mui/icons-material/AccountBox';

import Link from 'next/link';

const Profile = () => {

    return (

        <div className="max-w-md mx-auto px-4 py-8">

            <h1 className="text-2xl font-bold text-center mb-4">

                Your Profile

            </h1>

            <p className="mb-4 text-center">Welcome to your profile page.</p>

            <div className="border border-indigo-500 rounded p-10">

                <div className="flex gap-2">

                    <AccountBoxIcon />

                    <h2 className="text-lg font-semibold mb-2">

                        Personal Information

                    </h2>

                </div>

                <p>

                    Name: <span className="ms-2">John Smith</span>

                </p>

                <p>

                    Email:

                    <Link href="/" className="text-blue-400 inline-block ms-4">

                        tardis@gmail.com

                    </Link>

                </p>

                <p>

                    Phone:

                    <Link href="/" className="text-blue-400 ms-2">

                        95475949

                    </Link>

                </p>

            </div>

        </div>

    );

};

export default Profile;

**Login.tsx**

'use client';

import React, { useState } from 'react';

import Link from 'next/link';

const Login: React.FC = () => {

    const [email, setEmail] = useState('');

    const [password, setPassword] = useState('');

    const handleSubmit = (e: React.FormEvent<HTMLFormElement>) => {

        e.preventDefault();

    };

    return (

        <div className="flex justify-center mt-36">

            <div className="w-full max-w-md">

                <form

                    onSubmit={handleSubmit}

                    className="bg-white border border-indigo-500 rounded p-10 mb-4"

                >

                    <div className="mb-4">

                        <label

                            className="block text-gray-700 text-sm font-bold mb-2"

                            htmlFor="email"

                        >

                            Email

                        </label>

                        <input

                            id="email"

                            type="email"

                            placeholder="Email"

                            value={email}

                            onChange={(e) => setEmail(e.target.value)}

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

                        />

                    </div>

                    <div className="mb-4">

                        <label

                            className="block text-gray-700 text-sm font-bold mb-2"

                            htmlFor="password"

                        >

                            Password

                        </label>

                        <input

                            id="password"

                            type="password"

                            placeholder="Password"

                            value={password}

                            onChange={(e) => setPassword(e.target.value)}

                            className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

                        />

                    </div>

                    <div className="flex items-center justify-between">

                        <button

                            type="submit"

                            className="bg-blue-500 hover:bg-green-700 transition-color duration-500 text-white font-bold py-2 px-6 ms-auto me-auto rounded"

                        >

                            Login

                        </button>

                    </div>

                </form>

                <p className="text-center text-gray-500 text-xs">

                    Don&apos;t have an account?{' '}

                    <Link

                        href="/register"

                        className="text-blue-500 hover:text-blue-700"

                    >

                        Register

                    </Link>

                </p>

            </div>

        </div>

    );

};

export default Login;

**About.tsx**

'use client';

const About = () => {

    return (

        <div className="max-w-4xl mx-auto text-center px-4 py-8">

            <h1 className="text-3xl font-bold mb-4">

                About{' '}

                <span className="underline text-sky-400">ContactSphere</span>

            </h1>

            <div className="border border-indigo-300 p-4 rounded">

                <p className="mb-4">

                    ContactSphere is a web application built for managing

                    contacts, including storing, deleting, and editing them. We

                    are dedicated to ensuring data protection and privacy.

                </p>

                <p>

                    ContactSphere is an innovative web application crafted to

                    seamlessly connect people by managing a comprehensive

                    database of contacts. Our platform is designed to facilitate

                    the storing, deleting, and editing of contact information,

                    all while prioritizing the utmost data protection and

                    privacy. With a mobile-friendly interface, ContactSphere

                    guarantees an exceptional user experience across a variety

                    of devices, ensuring you stay connected effortlessly,

                    anytime and anywhere.

                </p>

            </div>

        </div>

    );

};

export default About;

**Server.js**

const express = require('express');

const fs = require('fs');

const app = express();

const port = 4000;

app.use((req, res, next) => {

    res.setHeader('Access-Control-Allow-Origin', 'http://localhost:3000');

    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE');

    res.setHeader('Access-Control-Allow-Headers', 'Content-Type');

    next();

});

app.use(express.json());

app.get('/contacts', (req, res) => {

    fs.readFile('contacts.json', 'utf8', (err, data) => {

        if (err) {

            console.error('Error reading contacts file:', err);

            res.status(500).send('Internal Server Error');

            return;

        }

        const contacts = JSON.parse(data);

        res.json(contacts);

    });

});

app.delete('/contacts/:id', (req, res) => {

    fs.readFile('contacts.json', 'utf8', (err, data) => {

        if (err) {

            console.error('Error reading contacts file:', err);

            return res.status(500).send('Internal Server Error');

        }

        let contacts = [];

        try {

            const jsonData = JSON.parse(data);

            contacts = jsonData.contacts;

        } catch (error) {

            console.error('Error parsing contacts JSON:', error);

            return res.status(500).send('Internal Server Error');

        }

        const id = req.params.id;

        contacts = contacts.filter((contact) => contact.id !== id);

        fs.writeFile(

            'contacts.json',

            JSON.stringify({ contacts }, null, 2),

            (err) => {

                if (err) {

                    console.error('Error writing to contacts file:', err);

                    return res.status(500).send('Internal Server Error');

                }

                res.status(204).end();

            }

        );

    });

});

app.post('/contacts', (req, res) => {

    fs.readFile('contacts.json', 'utf8', (err, data) => {

        if (err) {

            console.error('Error reading contacts file:', err);

            return res.status(500).send('Internal Server Error');

        }

        let contacts = [];

        try {

            const jsonData = JSON.parse(data);

            contacts = jsonData.contacts;

        } catch (error) {

            console.error('Error parsing contacts JSON:', error);

            return res.status(500).send('Internal Server Error');

        }

        const newContact = req.body;

        console.log('Received new contact:', newContact);

        if (!newContact.name || !newContact.phones) {

            return res.status(400).send('Name and phones are required');

        }

        newContact.id =

            contacts.length > 0

                ? (parseInt(contacts[contacts.length - 1].id) + 1).toString()

                : '1';

        contacts.push(newContact);

        fs.writeFile(

            'contacts.json',

            JSON.stringify({ contacts }, null, 2),

            (err) => {

                if (err) {

                    console.error('Error writing to contacts file:', err);

                    return res.status(500).send('Internal Server Error');

                }

                res.status(201).json(newContact);

            }

        );

    });

});

app.post('/contacts/reset', (req, res) => {

    fs.readFile('contacts.json', 'utf8', (err, data) => {

        if (err) {

            console.error('Error reading contacts file:', err);

            res.status(500).send('Internal Server Error');

            return;

        }

        let contacts = [];

        try {

            const jsonData = JSON.parse(data);

            contacts = jsonData.contacts;

        } catch (error) {

            console.error('Error parsing contacts JSON:', error);

            return res.status(500).send('Internal Server Error');

        }

        fs.writeFile(

            'contacts.json',

            JSON.stringify({ contacts }, null, 2),

            (err) => {

                if (err) {

                    console.error('Error writing to contacts file:', err);

                    return res.status(500).send('Internal Server Error');

                }

                res.json({ contacts });

            }

        );

    });

});

app.put('/contacts/:id', (req, res) => {

    const id = req.params.id;

    const updatedContact = req.body;

    fs.readFile('contacts.json', 'utf8', (err, data) => {

        if (err) {

            console.error('Error reading contacts file:', err);

            return res.status(500).send('Internal Server Error');

        }

        let contacts = [];

        try {

            const jsonData = JSON.parse(data);

            contacts = jsonData.contacts;

        } catch (error) {

            console.error('Error parsing contacts JSON:', error);

            return res.status(500).send('Internal Server Error');

        }

        const index = contacts.findIndex((contact) => contact.id === id);

        if (index === -1) {

            return res.status(404).send('Contact not found');

        }

        contacts[index] = { ...contacts[index], ...updatedContact };

        fs.writeFile(

            'contacts.json',

            JSON.stringify({ contacts }, null, 2),

            'utf8',

            (err) => {

                if (err) {

                    console.error('Error writing contacts file:', err);

                    return res.status(500).send('Internal Server Error');

                }

                res.json(contacts[index]);

            }

        );

    });

});

app.get('/contacts', (req, res) => {

    fs.readFile('contacts.json', 'utf8', (err, data) => {

        if (err) {

            console.error('Error reading contacts file:', err);

            res.status(500).send('Internal Server Error');

            return;

        }

        const contacts = JSON.parse(data);

        console.log(data);

        res.json(contacts);

    });

});

app.listen(port, () => {

    console.log(`Server is running at http://localhost:${port}`);

});

**Висновки**

В рамках лабораторної роботи була розроблена клієнтська частина веб-додатка "ContactSphere", яка повністю відповідає поставленим вимогам. Додаток забезпечує зручний інтерфейс для управління списком контактів, дозволяючи користувачам легко додавати нові контакти, редагувати та видаляти існуючі записи. Під час розробки використані сучасні технології, такі як Next.js, Tailwind CSS, Material Icons і JSON Server, що сприяє швидкому та ефективному розробленню додатка. Код дотримано по всім правильним стандартам, розділений на окремі компоненти, що полегшує його розуміння та підтримку. Інтерфейс додатка оптимізований для мобільних пристроїв, що робить його доступним для користувачів на різних пристроях і платформах. Отже, результат виконання лабораторної роботи відповідає поставленим завданням і дозволяє ефективно керувати списком контактів, використовуючи веб-інтерфейс.