Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №9
Технології розроблення програмного забезпечення
Різні види взаємодії додатків: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE
Особиста Бухгалтерія

Виконала:
Студентка групи ІА-22
Лахоцька С. О.

Перевірив:
Мягкий М. Ю

Київ 2024

Зміст

Тема: Різні види взаємодії додатків CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE
Мета: ознайомитись з теоретичними відомостями, реалізувати один з видів взаємодії між додатками.

Теоретичні відомості

Клієнт-серверна модель передбачає взаємодію між двома основними компонентами: клієнтом і сервером. Клієнт відповідає за взаємодію з користувачем і надсилання запитів до сервера, а сервер займається обробкою цих запитів, зберіганням даних та виконанням основної логіки. Залежно від розподілу функцій, клієнт може бути тонким або товстим. Тонкий клієнт відправляє більшість обчислювальних операцій на сервер, залишаючи собі лише відображення інформації, тоді як товстий виконує значну частину обчислень локально. Основною перевагою клієнт-серверної архітектури є централізоване управління, що спрощує адміністрування і зменшує ризик конфліктів даних. Проте високі навантаження на сервер і залежність від стабільності з'єднання можуть бути проблемою.

Однорангові мережі (peer-to-peer) побудовані на принципі рівноправності всіх учасників системи. У них немає центрального сервера, і клієнтські програми безпосередньо взаємодіють одна з одною для виконання спільних завдань. Основними викликами таких мереж є пошук клієнтів і синхронізація даних між ними. Для цього використовуються централізовані адресні списки або алгоритми обміну даними. Цей підхід добре підходить для децентралізованих систем, але може викликати труднощі у масштабуванні та забезпеченні стабільності зв'язку.

Сервіс-орієнтована архітектура (SOA) використовує модульний підхід, де програмне забезпечення розробляється як набір взаємодіючих служб. Кожна служба має стандартизований інтерфейс і виконує конкретну функцію. Основною перевагою є гнучкість, що дозволяє інтегрувати різні компоненти незалежно від їхніх платформ чи мов програмування. SOA широко використовується для побудови складних розподілених систем, таких як веб-сервіси або хмарні платформи. Однак розробка таких систем може бути складною через необхідність забезпечення сумісності між компонентами.

Модель SaaS (програмне забезпечення як послуга), яка ґрунтується на принципах SOA, дозволяє користувачам отримувати доступ до програм через Інтернет, спрощуючи управління і знижуючи витрати на обслуговування. Це економічно вигідно, оскільки користувач платить тільки за використання послуги, а не за володіння програмним забезпеченням.

Архітектура мікросервісів, яка є сучасною інтерпретацією SOA, дозволяє створювати серверні додатки як набір незалежних служб, кожна з яких

відповідає за виконання специфічної функції. Це забезпечує високу масштабованість і полегшує супроводження великих проектів, але вимагає належного рівня організації обміну даними між компонентами.

Хід роботи

Вона забезпечує чіткий поділ між інтерфейсом користувача та бізнес-логікою, що дозволяє ефективно масштабувати систему, розширювати функціонал та підтримувати високу продуктивність. Фронтенд реалізований на React, а бекенд — на Spring Boot, що дозволило швидко та зручно налаштувати взаємодію через REST API з використанням авторизації через JWT.

Архітектура SOA (Service-Oriented Architecture) не була використана з таких причин:

- Складність впровадження: SOA підходить для великих і складних систем, що складаються з багатьох незалежних модулів. Для проєкту з чітко визначеною клієнт-серверною взаємодією вона була б надмірною.

- Зайві витрати на організацію взаємодії сервісів: SOA вимагає стандартизованих протоколів і механізмів комунікації між службами, що збільшило б складність і час розробки без суттєвих переваг для такого масштабу проєкту.

- Відсутність необхідності в широкій інтеграції: У проєкті не передбачено інтеграції з іншими системами або платформами, для чого SOA є ідеальним рішенням. Усі компоненти додатку вже працюють у рамках однієї взаємодії (React і Spring Boot).

Таким чином, клієнт-серверна архітектура була оптимальним вибором для реалізації поставлених цілей, адже вона забезпечила простоту, надійність і високу швидкість розробки, уникнувши зайвої складності, яка супроводжувала б впровадження SOA.Клієнтська частина створена за допомогою бібліотеки React, що забезпечує сучасний і динамічний користувацький інтерфейс. Серверна частина реалізована за допомогою Spring Boot, що дозволяє швидко створювати веб-додатки на основі Java. Для обміну даними між клієнтом і сервером використовується REST API. Авторизація здійснюється за допомогою JWT-токенів, що забезпечує безпечний доступ до ресурсів.

Структуру клієнтського додатку зображено на рисунку 1.1.

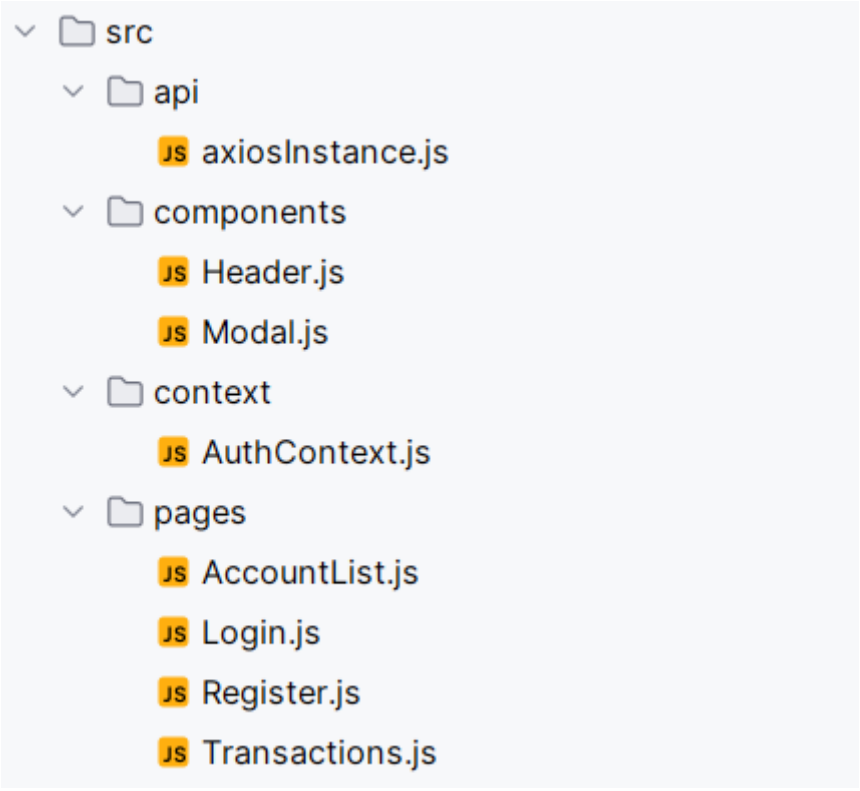

Рисунок 1.1 – Структура компонентів в клієнтському додатку

Контекст авторизації: Контекст AuthContext відповідає за управління станом авторизації (токеном) у локальному сховищі та перенаправлення після входу/виходу.

Компоненти інтерфейсу: Включають сторінки входу, реєстрації, список акаунтів, транзакцій та модальне вікно для створення транзакцій. Для передачі запитів до сервера використовується axios з налаштованими інтерцепторами для автоматичної вставки токенів.

Маршрутизація: Реалізована з використанням React Router, що дозволяє організувати навігацію між сторінками.

Приклад сторінки з даними про транзакції користувача на певному рахунку наведено на рисунку 1.2.

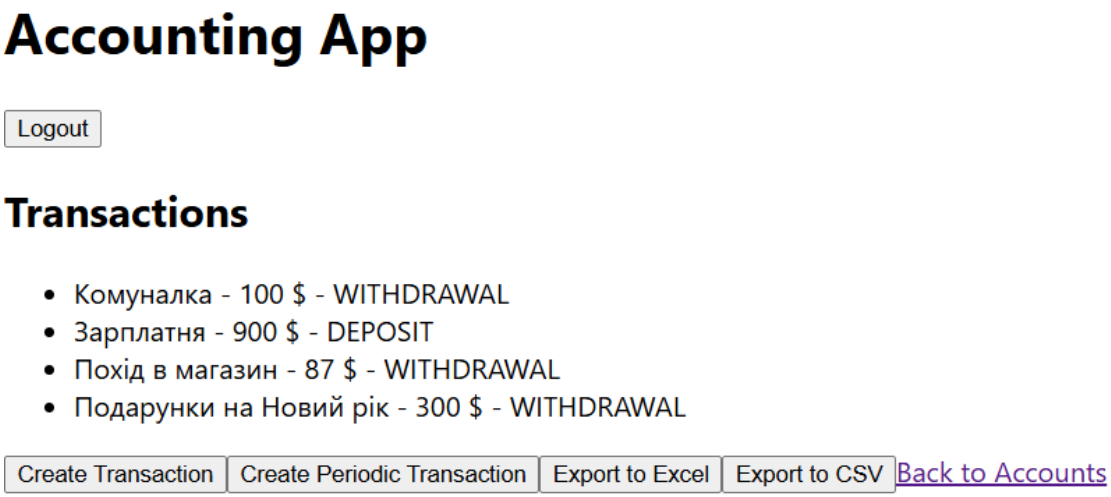

Рис 1.2. – Сторінка з транзакціями

Обрана клієнт-серверна архітектура забезпечує чіткий поділ відповідальностей між інтерфейсом користувача та бізнес-логікою. Це дозволяє легко масштабувати систему, розширювати функціонал і підтримувати високу продуктивність. Водночас використання сучасних технологій, таких як JWT, React і Spring Boot, робить проєкт надійним і зручним для розробки.

Висновки

У цій лабораторній роботі було реалізовано клієнт-серверну архітектуру для додатку особистої бухгалтерії. Завершено розробку системи. Було досліджено особливості та переваги використання інших типів архітектури, таких як P2P та SOA, що відкриває можливість їх використання в інших проектах, де це буде доцільно.

Вихідний код

```
// file: src/api/axiosInstance.js
// src/api/axiosInstance.js
import axios from 'axios';
import { useAuth } from '../context/AuthContext';

const useAxios = () => {
    const { token } = useAuth();

    const instance = axios.create({
```

```
        baseURL: 'http://localhost:8080',
    });

    instance.interceptors.request.use(
        (config) => {
            if (token) {
                config.headers.Authorization = `Bearer ${token}`;
            }
            return config;
        },
        (error) => Promise.reject(error)
    );

    return instance;
};

export default useAxios;


// file: src/pages/Login.js
import React, { useState } from 'react';
import { useAuth } from '../context/AuthContext';
import { useNavigate, Link } from 'react-router-dom';

const Login = () => {
    const [credentials, setCredentials] = useState({ username: '', password: ''
});
    const { login } = useAuth();
    const navigate = useNavigate();

    const handleSubmit = async (e) => {
        e.preventDefault();
        try {
            const response = await fetch('http://localhost:8080/auth/sign-in', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(credentials),
            });
            if (response.ok) {
                const data = await response.json();
                login(data.token, navigate); // Передаємо navigate
            } else {
                alert('Invalid username or password');
            }
        } catch (error) {
            console.error('Login error:', error);
        }
    };

    return (
        <div>
            <h2>Login</h2>
            <form onSubmit={handleSubmit}>
                <div>
                    <label>Username</label>
                    <input
                        type="text"
                        value={credentials.username}
                        onChange={(e) => setCredentials({ ...credentials,
username: e.target.value })}
                        required
                    />
                </div>
                <div>
                    <label>Password</label>
```

```
                    <input
                        type="password"
                        value={credentials.password}
                        onChange={(e) => setCredentials({ ...credentials,
password: e.target.value })}
                        required
                    />
                </div>
                <button type="submit">Login</button>
            </form>
            <div style={{ marginTop: '10px' }}>
                <p>Don't have an account?</p>
                <Link to="/register">
                    <button type="button">Register</button>
                </Link>
            </div>
        </div>
    );
};

export default Login;


// file: src/pages/Register.js
import React, { useState } from 'react';

const Register = () => {
    const [formData, setFormData] = useState({
        username: '',
        email: '',
        password: '',
    });

    const [error, setError] = useState('');
    const [success, setSuccess] = useState(false);

    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData({ ...formData, [name]: value });
    };

    const handleSubmit = async (e) => {
        e.preventDefault();
        setError('');
        setSuccess(false);

        try {
            const response = await fetch('http://localhost:8080/auth/sign-up', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(formData),
            });

            if (response.ok) {
                setSuccess(true); // Показуємо повідомлення про успіх
            } else {
                const errorData = await response.json();
                setError(errorData.message || 'Failed to register');
            }
        } catch (err) {
            setError('Something went wrong. Please try again.');
        }
    };

    return (
```

```
        <div>
            <h2>Register</h2>
            {success && <p style={{ color: 'green' }}>Registration successful!
You can now log in.</p>}
            {error && <p style={{ color: 'red' }}>{error}</p>}
            <form onSubmit={handleSubmit}>
                <div>
                    <label>Username:</label>
                    <input
                        type="text"
                        name="username"
                        value={formData.username}
                        onChange={handleChange}
                        required
                    />
                </div>
                <div>
                    <label>Email:</label>
                    <input
                        type="email"
                        name="email"
                        value={formData.email}
                        onChange={handleChange}
                        required
                    />
                </div>
                <div>
                    <label>Password:</label>
                    <input
                        type="password"
                        name="password"
                        value={formData.password}
                        onChange={handleChange}
                        required
                    />
                </div>
                <button type="submit">Register</button>
            </form>
        </div>
    );
};

export default Register;


// file: src/pages/AccountList.js
import React, { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import Modal from '../components/Modal';
import useAxios from '../api/axiosInstance';

const currencySymbols = {
    '840': '$', // USD
    '978': '€', // EUR
    '826': '£', // GBP
    '392': '¥'  // JPY
};

const codeToSymbol = (code) => currencySymbols[code] || code;
const symbolToCode = (symbol) => {
    return Object.keys(currencySymbols).find(key => currencySymbols[key] ===
symbol) || symbol;
};

const AccountList = () => {
```

```
    const [accounts, setAccounts] = useState([]);
    const [isModalOpen, setIsModalOpen] = useState(false);
    const [newAccount, setNewAccount] = useState({
        name: '',
        description: '',
        balance: 0,
        currencyCode: '840'
    });
    const [error, setError] = useState('');
    const axios = useAxios();

    // Fetch accounts
    const fetchAccounts = async () => {
        try {
            const response = await axios.get('/accounts');
            setAccounts(response.data);
        } catch (error) {
            console.error('Error fetching accounts', error);
        }
    };

    // Create new account
    const createAccount = async (e) => {
        e.preventDefault();
        try {
            const requestData = {
                ...newAccount,
                currencyCode: parseInt(newAccount.currencyCode) // Ensure
numeric code
            };
            const response = await axios.post('/accounts', requestData);
            setAccounts([...accounts, response.data]);
            setNewAccount({ name: '', description: '', balance: 0, currencyCode:
'840' });
            setIsModalOpen(false);
        } catch (error) {
            setError('Failed to create account. Please try again.');
            console.error('Error creating account', error);
        }
    };

    useEffect(() => {
        fetchAccounts();
    }, []);

    return (
        <div>
            <h2>Accounts</h2>
            <ul>
                {accounts.map((account) => (
                    <li key={account.id}>
                        <Link to={`/accounts/${account.id}/transactions`}>
                            <strong>{account.name}</strong> -
{account.description}
                        </Link>
                        <div>
                            <p>Balance: {account.balance}
{codeToSymbol(account.currencyCode)}</p>
                            <p>User: {account.user}</p>
                        </div>
                    </li>
                ))}
            </ul>
```

```
            <button onClick={() => setIsModalOpen(true)}>Create New
Account</button>

            <Modal isOpen={isModalOpen} onClose={() => setIsModalOpen(false)}>
                <h3>Create New Account</h3>
                {error && <p style={{ color: 'red' }}>{error}</p>}
                <form onSubmit={createAccount}>
                    <div>
                        <label>Name:</label>
                        <input
                            type="text"
                            value={newAccount.name}
                            onChange={(e) => setNewAccount({ ...newAccount,
name: e.target.value })}
                            required
                        />
                    </div>
                    <div>
                        <label>Description:</label>
                        <input
                            type="text"
                            value={newAccount.description}
                            onChange={(e) => setNewAccount({ ...newAccount,
description: e.target.value })}
                            required
                        />
                    </div>
                    <div>
                        <label>Balance:</label>
                        <input
                            type="number"
                            value={newAccount.balance}
                            onChange={(e) => setNewAccount({ ...newAccount,
balance: parseFloat(e.target.value) })}
                            required
                        />
                    </div>
                    <div>
                        <label>Currency:</label>
                        <select
                            value={newAccount.currencyCode}
                            onChange={(e) => setNewAccount({ ...newAccount,
currencyCode: e.target.value })}
                            required
                        >
                            {Object.entries(currencySymbols).map(([code,
symbol]) => (
                                <option key={code}
value={code}>{symbol}</option>
                            ))}
                        </select>
                    </div>
                    <button type="submit">Create Account</button>
                </form>
            </Modal>
        </div>
    );
};

export default AccountList;


// file: src/pages/Transactions.js
import React, { useState, useEffect } from 'react';
import { useParams, Link } from 'react-router-dom';
```

```
import useAxios from '../api/axiosInstance';
import Modal from '../components/Modal';

const currencySymbols = {
    '840': '$', // USD
    '978': '€', // EUR
    '826': '£', // GBP
    '392': '¥'  // JPY
};

const codeToSymbol = (code) => currencySymbols[code] || code;

const Transactions = () => {
    const { accountId } = useParams();
    const [transactions, setTransactions] = useState([]);
    const [currencyCode, setCurrencyCode] = useState('840');
    const [newTransaction, setNewTransaction] = useState({
        name: '',
        description: '',
        amount: 0,
        transactionType: 'WITHDRAWAL'
    });
    const [newPeriodicTransaction, setNewPeriodicTransaction] = useState({
        name: '',
        description: '',
        amount: 0,
        period: 'DAILY',
        transactionType: 'WITHDRAWAL'
    });
    const [isModalOpen, setIsModalOpen] = useState(false);
    const [isPeriodicModalOpen, setIsPeriodicModalOpen] = useState(false);
    const axios = useAxios();

    const fetchAccountDetails = async () => {
        try {
            const response = await axios.get(`/accounts/${accountId}`);
            setCurrencyCode(response.data.currencyCode);
        } catch (error) {
            console.error('Error fetching account details', error);
        }
    };

    const fetchTransactions = async () => {
        try {
            const response = await
axios.get(`/accounts/${accountId}/transactions`);
            setTransactions(response.data);
        } catch (error) {
            console.error('Error fetching transactions', error);
        }
    };

    const createTransaction = async () => {
        try {
            const response = await
axios.post(`/accounts/${accountId}/transactions`, newTransaction);
            setTransactions([...transactions, response.data]);
            setIsModalOpen(false);
        } catch (error) {
            console.error('Error creating transaction', error);
        }
    };

    const createPeriodicTransaction = async () => {
        try {
```

```
            const response = await
axios.post(`/accounts/${accountId}/transactions/period`,
newPeriodicTransaction);
            setIsPeriodicModalOpen(false);
        } catch (error) {
            console.error('Error creating periodic transaction', error);
        }
    };

    const exportTransactions = async (format) => {
        try {
            const response = await
axios.get(`/accounts/${accountId}/statistics`, {
                params: { exportType: format },
                responseType: 'blob'
            });
            const url = window.URL.createObjectURL(new Blob([response.data]));
            const link = document.createElement('a');
            link.href = url;
            link.setAttribute('download', `transactions.${format}`);
            document.body.appendChild(link);
            link.click();
        } catch (error) {
            console.error(`Error exporting transactions as ${format}`, error);
        }
    };

    useEffect(() => {
        fetchAccountDetails();
        fetchTransactions();
    }, [accountId]);

    return (
        <div>
            <h2>Transactions</h2>
            <ul>
                {transactions.map(transaction => (
                    <li key={transaction.id}>
                        {transaction.name} - {transaction.amount}
{codeToSymbol(currencyCode)} - {transaction.transactionType}
                    </li>
                ))}
            </ul>
            <button onClick={() => setIsModalOpen(true)}>Create
Transaction</button>
            <button onClick={() => setIsPeriodicModalOpen(true)}>Create Periodic
Transaction</button>
            <button onClick={() => exportTransactions('XLS')}>Export to
Excel</button>
            <button onClick={() => exportTransactions('CSV')}>Export to
CSV</button>

            <Modal isOpen={isModalOpen} onClose={() => setIsModalOpen(false)}>
                <h3>Create Transaction</h3>
                <div style={{ display: 'flex', flexDirection: 'column', gap:
'10px' }}>
                    <label>
                        Transaction Name:
                        <input
                            type="text"
                            value={newTransaction.name}
                            onChange={e => setNewTransaction({
...newTransaction, name: e.target.value })}
                        />
                    </label>
```

```
                    <label>
                        Amount:
                        <input
                            type="number"
                            value={newTransaction.amount}
                            onChange={e => setNewTransaction({
...newTransaction, amount: parseFloat(e.target.value) })}
                        />
                    </label>
                    <label>
                        Transaction Type:
                        <select
                            value={newTransaction.transactionType}
                            onChange={e => setNewTransaction({
...newTransaction, transactionType: e.target.value })}
                        >
                            <option value="WITHDRAWAL">Withdrawal</option>
                            <option value="DEPOSIT">Deposit</option>
                        </select>
                    </label>
                </div>
                <button onClick={createTransaction}>Add Transaction</button>
            </Modal>

            <Modal isOpen={isPeriodicModalOpen} onClose={() =>
setIsPeriodicModalOpen(false)}>
                <h3>Create Periodic Transaction</h3>
                <div style={{ display: 'flex', flexDirection: 'column', gap:
'10px' }}>
                    <label>
                        Name:
                        <input
                            type="text"
                            value={newPeriodicTransaction.name}
                            onChange={e => setNewPeriodicTransaction({
...newPeriodicTransaction, name: e.target.value })}
                        />
                    </label>
                    <label>
                        Amount:
                        <input
                            type="number"
                            value={newPeriodicTransaction.amount}
                            onChange={e => setNewPeriodicTransaction({
...newPeriodicTransaction, amount: parseFloat(e.target.value) })}
                        />
                    </label>
                    <label>
                        Period:
                        <select
                            value={newPeriodicTransaction.period}
                            onChange={e => setNewPeriodicTransaction({
...newPeriodicTransaction, period: e.target.value })}
                        >
                            <option value="MINUTELY">Minutely</option>
                            <option value="HOURLY">Hourly</option>
                            <option value="DAILY">Daily</option>
                            <option value="WEEKLY">Weekly</option>
                            <option value="MONTHLY">Monthly</option>
                            <option value="YEARLY">Yearly</option>
                        </select>
                    </label>
                    <label>
                        Transaction Type:
                        <select
```

```
                            value={newPeriodicTransaction.transactionType}
                            onChange={e => setNewPeriodicTransaction({
...newPeriodicTransaction, transactionType: e.target.value })}
                        >
                            <option value="WITHDRAWAL">Withdrawal</option>
                            <option value="DEPOSIT">Deposit</option>
                        </select>
                    </label>
                </div>
                <button onClick={createPeriodicTransaction}>Add Periodic
Transaction</button>
            </Modal>

            <Link to="/">Back to Accounts</Link>
        </div>
    );
};

export default Transactions;


// file: src/App.js
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import AccountList from './pages/AccountList';
import Transactions from './pages/Transactions';
import Header from "./components/Header";
import Login from "./pages/Login";
import Register from './pages/Register'; // Імпортуємо компонент реєстрації

function App() {
    return (
        <Router>
            <div className="App">
                <Header />
                <Routes>
                    <Route path="/" element={<AccountList />} />
                    <Route path="/login" element={<Login />} />
                    <Route path="/accounts/:accountId/transactions"
element={<Transactions />} />
                    <Route path="/register" element={<Register />} /> {/* Новий
маршрут */}
                </Routes>
            </div>
        </Router>
    );
}

export default App;


// file: src/App.css
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
```

```
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

```
// file: src/context/AuthContext.js
import React, { createContext, useState, useContext } from 'react';

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
    const [token, setToken] = useState(localStorage.getItem('jwtToken') || '');

    const login = (newToken, navigate) => {
        setToken(newToken);
        localStorage.setItem('jwtToken', newToken);
        navigate('/'); // Викликаємо передану функцію для редіректу
    };

    const logout = (navigate) => {
        setToken('');
        localStorage.removeItem('jwtToken');
        navigate('/login'); // Викликаємо передану функцію для редіректу
    };

    return (
        <AuthContext.Provider value={{ token, login, logout }}>
            {children}
        </AuthContext.Provider>
    );
};

export const useAuth = () => useContext(AuthContext);
```

```
// file: src/index.js
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { AuthProvider } from './context/AuthContext';
```

```
const rootElement = document.getElementById('root'); // Знайти контейнерний
елемент

if (rootElement) {
    const root = ReactDOM.createRoot(rootElement); // Створити корінь React
    root.render(
        <React.StrictMode>
            <AuthProvider>
                <App />
            </AuthProvider>
        </React.StrictMode>
    );
} else {
    console.error("Root element not found. Please check your index.html.");
}

// file: src/index.css
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}


// file: src/components/Modal.js
function Modal({ isOpen, onClose, children }) {
    if (!isOpen) return null;

    return (
        <div style={styles.overlay}>
            <div style={styles.modal}>
                <button style={styles.closeButton} onClick={onClose}>
                    &times;
                </button>
                {children}
            </div>
        </div>
    );
}

const styles = {
    overlay: {
        position: 'fixed',
        top: 0,
        left: 0,
        width: '100%',
        height: '100%',
        backgroundColor: 'rgba(0, 0, 0, 0.5)',
        display: 'flex',
        justifyContent: 'center',
        alignItems: 'center',
        zIndex: 1000,
    },
    modal: {
        backgroundColor: '#fff',
        padding: '20px',
```

```
        borderRadius: '8px',
        width: '400px',
        position: 'relative',
    },
    closeButton: {
        position: 'absolute',
        top: '10px',
        right: '10px',
        fontSize: '18px',
        border: 'none',
        background: 'none',
        cursor: 'pointer',
    },
};

export default Modal


// file: src/components/Header.js
import React from 'react';
import { useAuth } from '../context/AuthContext';
import { useNavigate, useLocation } from 'react-router-dom';

const Header = () => {
    const { logout } = useAuth();
    const navigate = useNavigate(); // Додаємо useNavigate
    const location = useLocation(); // Додаємо useLocation для перевірки шляху

    return (
        <header>
            <h1>Accounting App</h1>
            {(location.pathname !== '/login' && location.pathname !==
'/register') && (
                <button onClick={() => logout(navigate)}>Logout</button> //
Передаємо navigate
            )}
        </header>
    );
};

export default Header;


// file: src/App.test.js
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});


// file: src/setupTests.js
// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom
import '@testing-library/jest-dom';


// file: src/reportWebVitals.js
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
```

```
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```