МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)

червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп’ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна система управління відносинами з клієнтами у сфері надання послуг»

здобувачки групи ІН – 91 Листопад Поліна Сергіївна.

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Поліна ЛИСТОПАД.
(підпис)

Керівник,
асистент кафедри комп’ютерних наук, кандидат фізико-математичних наук Олександр ВЛАСЕНКО. _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувачки групи ІН-91 Листопад П.С.

1. Тема роботи: «Інформаційна система управління відносинами з клієнтами у сфері надання послуг»
затверджую наказом по СумДУ від _____
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для створення інформаційних систем управління відносинами з клієнтами. 3) Розробка клієнтської частини інформаційної системи управління відносинами з клієнтами 4) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Аналіз проблеми предметної області, постановка й формування завдань дослідження		
2	Огляд технологій, що використовуються для створення систем управління відносинами з клієнтами		
3	Розробка клієнтської частини інформаційної системи управління відносинами з клієнтами		
4	Аналіз отриманих результатів		
5	Оформлення пояснювальної записки до кваліфікаційної роботи		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 55 стр., 10 рис., 1 додаток, 27 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки дозволяє підприємствам організовувати роботу компанії та управляти відносинами з клієнтами за допомогою інформаційної системи.

Об’єкт дослідження — система управління відносинами з клієнтами.

Мета роботи — розробка інформаційної системи управління відносинами з клієнтами за допомогою фреймворку React JS.

Методи дослідження — інструменти створення клієнтської частини інформаційних систем.

Результати — розроблено клієнтську частину інформаційної системи для управління відносинами з клієнтами в сфері надання послуг, яка дозволяє відображати дані про клієнтів, вводити дані нових клієнтів, створювати та переглядати події в календарі та створювати список справ.

ІНФОРМАЦІЙНА СИСТЕМА, УПРАВЛІННЯ ВІДНОСИНАМИ З КЛІЄНТАМИ, CRM, REACT JS.

ЗМІСТ

ВСТУП

Більшості складно уявити своє життя без комп’ютерів, смартфонів та інших технологій. І це дуже легко зрозуміти, оскільки вони стали невід’ємною частиною нашого життя. Їх вплив неможливо оцінити. Вони дають людям майже необмежені можливості. Кількість додатків зростає з кожним днем. І хоча не завжди можна знайти саме такий додаток, який необхідний для певних завдань, технології, що існують, дозволяють створити такі додатки самостійно.

Різні технології використовуються у всіх сферах життя. Це ігри, в які ми граємо заради свого задоволення та розваг. Це також державні додатки, що полегшують життя громадян при виконанні складних щоденних операцій та сприяють комунікації з представниками влади. Це також банківські системи для полегшення щоденних операцій, інтернет магазини для будь-яких товарів, додатки для навчання в університетах, школах та індивідуального розвитку, додатки для розвитку та розваг дітей.

Не варто забувати і про соціальні мережі, кількість яких зростає з кожним днем та які стали важливою частиною нашого спілкування. Ними користуються близько 4,14 мільярдів людей, що складає більше половини населення світу[1]. Такі програми як Facebook, Instagram, Twitter, Telegram та багато інших, допомагають нам підтримувати зв’язок з рідними та друзями на величезних відстанях, знаходити нових друзів за спільними інтересами, слідкувати за улюбленими акторами та співаками, дізнаватися новини з усього світу та пізнавати цей світ не полишаючи власної кімнати.

Ще однією перевагою додатків є те, що їх використання дозволяє зберегти час, уникнути помилок та організувати роботу. Серед таких додатків можна знову привести у приклад банківські системи, що значно зберігають час людини. Для компанії перевага у часі й організації дуже гарно представлена у додатках типу CRM, тобто у системах управління

відносинами з клієнтами. Багато компаній створюють подібні додатки, щоб надавати свої послуги ще зручніше, швидше та зрозуміліше, ставлячи своїх клієнтів на перше місце.

Назвати можна ще більше переваг технологій, додатків та інформаційний систем, але навіть лише з наведених вище аргументів зрозуміло, що такі продукти є важливою частиною життя людини, а отже є актуальними як ніколи. Великий попит на них показує, що створення таких додатків необхідне та користується неабиякою популярністю. Впевнено можна сказати, що їх актуальність лише зростатиме з часом.

Мета роботи полягає у тому, щоб проаналізувати які продукти можна створювати за допомогою фреймворку React JS, дізнатися про особливості його роботи та процеси, які при цьому відбуваються, з’ясувати, які бібліотеки використовуються для створення систем управління. Кінцевим результатом та показником рівня володіння темою є створення інформаційної системи управління відносинами з клієнтами у сфері послуг, що використовує фреймворк React JS, а також проведений аналіз процесу створення та отриманих результатів.

Робота складається зі вступу, аналітичного огляду, постановки задачі, вибору методів розв’язання задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз принципів роботи

Більшість великих компаній мають свої системи для організації роботи. Багато IT компаній використовують додатки для планування та керування проєктами, такі як Jira, Trello, ClickUp, Binfire, Basecamp[2]. Для організації роботи великої кількості працівників необхідно мати систему, що буде містити спільну інформацію, доступну у будь-який момент. Проте для невеликих підприємств часто важко знайти такий інструмент, що буде задовольняти вимоги та підходити функціонально та фінансово. Конкуренція постійно зростає і для того, щоб змагатися з великими компаніям, невеликим бізнесам особливо важливо зберігати надійні та довірливі стосунки з своїми клієнтами[3]. Одним із способів досягання цього є системи управління відносинами з клієнтами у сфера надання послуг. Наразі їх створення є актуальним та необхідним через їх позитивний вплив та переваги.

CRM, або Customer Relationship Management (в перекладі система управління взаємовідносинами з клієнтами) – це тип додатків, що призначені для організації та автоматизації взаємовідносин з клієнтами. Мета такої системи – оптимізувати роботу працівників та задовільнити потреби клієнта[4]. Такі системи допомагають компаніям надавати високу якість обслуговування користувачам та контролювати складні процеси в єдиному місці.

CRM зазвичай передбачає такі три принципи:

1) Подібна система повинна мати спільну загальну систему інформації, доступ до якої є будь-який час, для того, щоб отримати дані про клієнтів.

2) Синхронізація каналів зв’язку, що використовує певні правила та процедури, які контролюють застосування такої системи в підприємстві.

3) Постійний збір та аналіз отриманих даних про користувачів для подальшого прийняття рішень, прогнозування, тощо[5].

Серед можливостей використання подібних систем підприємствами варто зазначити наступні:

1) Швидке отримання актуальних даних про клієнтів, звітних даних, документів.

2) Управління бізнес-процесами компанії та контроль роботи звичайних працівників, менеджерів.

3) Планування та управління продажами, маркетингом, фінансами та іншими економічними процесами[6].

Прикладами успішних CRM систем, що активно використовується у світі та особливо в Україні є:

1) Microsoft Dynamis CRM.
2) Oracle Siebel CRM.
3) Agile CRM.
4) PERFECTUM CRM.
5) SAP.
6) Zoho CRM.
7) ONEBOX CRM.
8) TEAMWOX.
9) BLOCKNOTAPP.
10) APPTIVO CRM SYSTEM[7].

Більшість з них використовується в IT сфері, проте деякі використовуються також для медичної сфери, сфери послуг, фінансових організацій, страхових компаній, тощо. Такі системи мають приблизно однаковий функціонал, що дозволяє надсилати сповіщення клієнтам, нагадувати про зустрічі, робити прогнозування, використовувати штучний інтелект для покращення роботи, створювати завдання для всієї команди, проводити фінансовий облік та багато іншого. Хоча системи і мають схожі

функції, проте їх кількість є виправданою. CRM завжди має бути гнучкою, щоб задовільнити потреби багатьох різноманітних підприємств. Але хоч як би одна система не намагалася включити в себе всі можливі функції, іноді інша система просто є зручнішою для певних цілей певного підприємства. Іноді компанії створюють подібні додатки окремо для себе, таким чином забезпечуючи наявність всіх необхідних функцій.

При створенні систем управління відносинами з клієнтами у сфері послуг варто згадати і про проблеми, які можуть виникнути у подальшому використані подібних систем. По-перше, робота з даними клієнта передбачає зберігання конфіденційної інформації, тому важливо пам’ятати про безпеку клієнтів та запровадити інструменти для захисту даних. Конфіденційність людини – це не тільки бажання, а й необхідність, право. Тому це також має бути пріоритетом компанії, адже довіра клієнта – головне. Також при деяких технічних проблем може виникнути можливість втрати даних. Для того, щоб уникнути цієї ситуації або повернутися до роботи з мінімальними втратами, система повинна бути готова відновити втрачену інформацію так, щоб не зашкодити ні підприємству, ні клієнту.

По-друге, велика кількість інструментів, що присутня в системах управління відносинами з клієнтами, часто призводить до того, що додаткові ресурси мають бути використані для навчання працівників керуванню подібною системою. Для того, щоб зменшити кількість цих ресурсів необхідно створити такий додаток, що є інтуїтивно зрозумілим та вкладати в нього такі інструменти, які справді будуть необхідні для роботи. Нагромадження багатьох функцій, що навіть ніколи не будуть використанні тільки завадить роботі та збільшить час на навчання працівників.

Також при введенні такої системи у використання компанією варто подумати про час та вартість адміністративного керування системою. До цього входить і навчання персоналу, кошти на імплементацію системи, підтримка програмного забезпечення, тощо. Розробка програми ніколи не

закінчується після випуску програми. Для нормальної роботи програми, її необхідно постійно підтримувати та оновлювати, що також займає часові та фінансові ресурси[8].

CRM – це не тільки система, яка допомагає організувати роботу підприємства, але й також бізнес стратегія, що передусім піклується про клієнта. Така модель дозволяє зберігати постійних користувачів та збільшувати доходи компанії у майбутньому. Тому без сумнівів можна сказати, що актуальність таких систем лише зростатиме з часом.

1.2 Постановка задачі

Для кваліфікаційної роботи необхідно створити інформаційну систему, що використовуватиметься для управління відносинами з клієнтами у сфері надання послуг за допомогою технології фреймворку React JS та додаткових інструментів. Для виконання роботи необхідно реалізувати наступні задачі:

1. Дослідити можливості фреймворку React JS та проаналізувати його використання для створення інформаційної системи управління відносинами з клієнтами.
2. Виконати огляд інструментів, що будуть додатково використані для розробки компонентів, таких як календар, тощо.
3. Розробити вимоги до додатка, використовуючи знання про принципи побудови інформаційних систем управління відносинами з клієнтами у сфері послуг та приклади конкурентних програм.
4. Розробити інтуїтивно зрозумілий інтерфейс додатка, використовуючи правила та принципи створення дизайну інформаційних систем та перевірити його відповідність зазначеним вимогам.
5. Розробити алгоритм роботи створення інформаційної системи управління відносинами з клієнтами у сфері надання послуг за допомогою обраних інструментів.

6. Розробити клієнтську частину інформаційної системи, що відповідає вимогам та створеному прототипу, використовуючи обрані інструменти та технології.
7. Сформулювати та викласти висновки щодо виконаної роботи.

При успішній реалізації системи буде створено клієнтську частину інформаційної системи, що допомагатиме невеликим підприємствам керувати відносинами з клієнтами, відображати дані про них, контролювати свою завантаженість та створювати задачі.

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Фреймворк React

Більшість сучасних додатків та веб-сайтів використовує JavaScript для того, щоб зробити свої продукти швидшими, більш інтерактивними для зручнішими для використання. Вимоги користувачів ростуть з кожним днем і необхідне рішення, яке допоможе задовільнити їх потреби та надати додаткам можливість конкурувати з іншими ресурсами. На відміну від звичайних традиційних сторінок, що витрачають багато часу на завантаження кожної сторінки при будь-яких невеликих змінах, модель розробки односторінкової програми, що використовується у JavaScript, дозволяє змінювати лише той контент на сторінці, до якого застосовуються зміни. Додатки з такою технологією вже давно набули популярності та зараз активно використовуються. Існує дуже велика кількість фреймворків, що робить це реальним. Найпопулярнішими серед них є Angular, React JS, React Native та Vue JS[9].

Для створення проєкту вирішено використовувати фреймворк React JS.

React JS – це фреймворк JavaScript, який був створений Джорданом Волком, працівником компанії Facebook. Це проєкт з відкритим початковим кодом, що був представлений у 2011 році, а згодом був застосований до проєктів компанії[10]. В 2011 в Facebook розробники почали стикатися з проблемами в організації та обслуговуванням коду. Зі збільшенням нових функцій та появою нових робітників швидкість діяльності додатків почала зменшуватись. Виконувати роботу ставало ще важче, а новий код накопичувався так, що розробники не встигали за оновленнями. Рішенням цим проблем стала бібліотека React JS. Вже в 2012 Facebook та Instagram почали використовувати нову технологію, а вже через кілька років React JS використовувався у інших додатках, серед яких Netflix, Airbnb та інші. Спочатку багато людей дуже скептично ставилися до новітньої розробки, але

фреймворк своїми результатами довів свою ефективність та зараз є дуже популярним[11].

В JavaScript також існує фреймворк, який має схожу назву – React Native. Через цю схожість деякі люди можуть плутати його з React JS. Насправді це два різні фреймворки. Обидва розроблені Facebook та активно використовуються у розробці додатків. До того ж, React Native – це бібліотека, що базувалася на React JS. Різниця між ними полягає у тому, що в той час як React JS використовується для побудови веб-додатків, React Native використовується для створення мобільних додатків для Android, iOS та Windows. Перший використовує віртуальну об'єктну модель документу для відтворення коду в браузері, в той час як другий використовує Native API для відтворення компонентів[12].

В деяких джерелах також можна побачити просто назви React та React JS. В цьому випадку, це означає майже те ж саме.

Щодо деяких особливостей React JS важливо згадати ще те, що синтаксис, що використовується в даному фреймворку, називається JSX. Це суміш HTML та JavaScript з повною потужністю останнього, що використовується під час написання веб додатків. Він використовуються для створення елементів, що складають компоненти.

Оскільки інформаційна система управління відносинами з клієнтами у сфері послуг буде будуватися як веб-додаток, використовуватиметься фреймворк React JS.

2.2 Середовище програмування

Для створення та редагування коду обрано інтегроване середовище програмування Visual Studio Code, або скороченоVS Code (Рисунок 2.1). Це середовище програмування було створено Microsoft в 2015. Воно підтримує багато різних мов програмування, серед яких C, C++, JavaScript, Python, Node.js, тощо[28]. Його інтерфейс дуже простий та інтуїтивно зрозумілий.

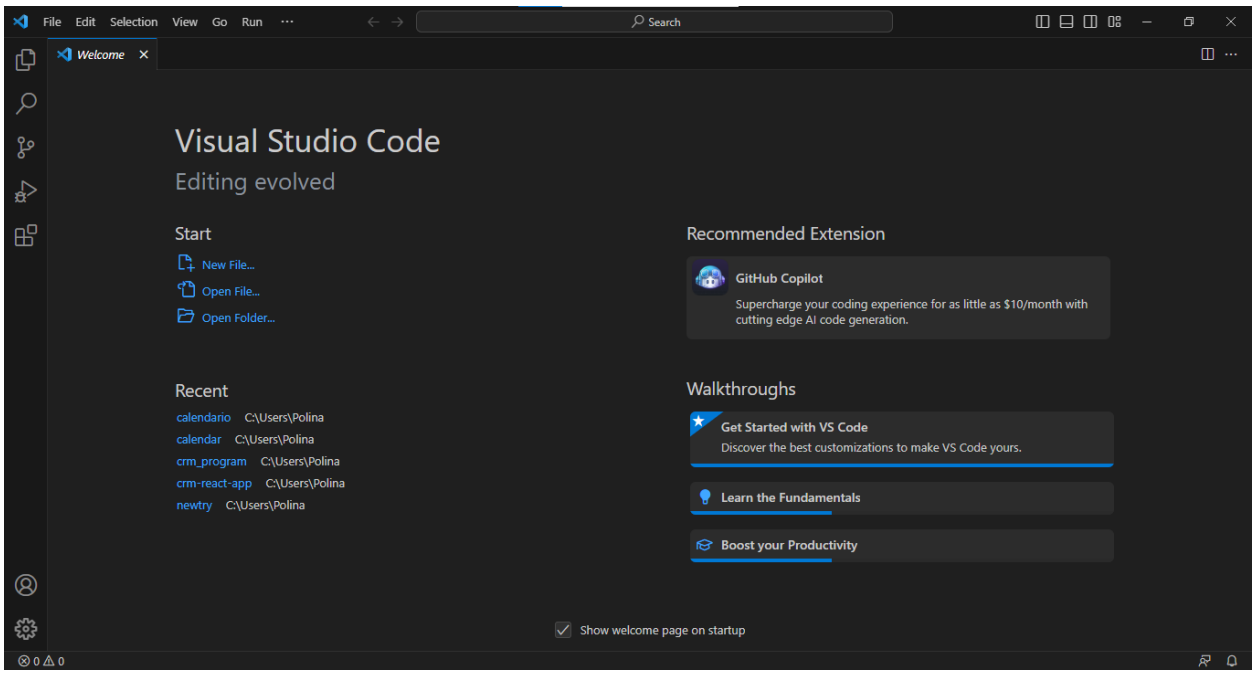

Рисунок 2.1 – Visual Studio Code

VS Code - це більш легка версія середовища Visual Studio. Він дозволяє користувачам працювати на будь-яких операційних системах: Windows, Linux та Mac. Користувач може писати, компілювати та запускати написані програми, працювати з різноманітними розширеннями для полегшення роботи, додавати нові теми до середовища, змінювати мову середовища, використовувати додаткові сервіси. До середовища також можна додати систему контролю версій, наприклад Git, що оптимізує роботу з проєктом. Під час написання коду, VS Code також пропонує можливі варіанти команд та допомагає з пошуком змінних, розпізнає та підсвічує рядки коду, вказує на можливі помилки та структурує зміст документу[14].

Для створення клієнтської сторони інформаційної системи це середовище дуже підходить, оскільки воно підтримує обрану мову програмування – JavaScript, містить багато необхідних інструментів для створення необхідних компонентів та функцій, має визначену структуру та є зручним в користуванні. На відміну від інших середовищ VS Code є значно швидшим та загалом найкращим вибором при веб розробці.

2.3 Бібліотека Syncfusion

Syncfusion – це бібліотека компонентів інтерфейсу користувача, що дозволяє використовувати високо функціональні та адаптивні компоненти в реалізації програми. Ця бібліотека має багато переваг і є значно кращою за альтернативи, такі як FullCalendar, Multiple UI, тощо. Компоненти, що присутні у Syncfusion, побудовані як модулі, що дозволяє використовувати їх вибірково та застосовувати необхідні функції. Вони є адаптивними, тому з легкістю можуть бути використані для додатків як на комп'ютері, так і на звичайних телефонах. Також вони постійно розвиваються та дороблюються розробниками для досягання найкращого користувацького досвіду та зовнішнього вигляду[15]. Також вони є дуже легкими для використання в коді і не потребують додаткових розширень чи інструментів для їх імплементації.

Найпопулярнішими компонентами бібліотеки Syncfusion є:

1) DataGrid – компонент, що дозволяє відображати та управляти даними на сторінці, а також експортувати інформацію з певних файлів.
2) Charts – компонент, що допомагає відображати інформацію за допомогою діаграм.
3) ListView – використовує заготовлені шаблони для відображення та контролю даних.
4) Scheduler – компонент, що дозволяє управляти подіями та даними.
5) Diagram – компонент, що дозволяє створювати різноманітні діаграми та схеми.

Подібні компоненти використовують дуже багато компаній на своїх сторінка, тому не дивно, що дана бібліотека є дуже популярною. Серед компаній, що користуються бібліотекою Syncfusion найвідомішими є Apple, IBM, Siemens, Bosch, тощо[16].

Для реалізації сторінки календаря в інформаційній системі управління відносинами з клієнтами у сфері послуг використовується компонент Scheduler. Це компонент надає можливість користувачу за допомогою програми створювати, змінювати, видаляти та відображати події, необхідні йому.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вимоги до проєкту

Перед розробкою будь-якого проєкту спочатку варто визначити його вимоги. Вимоги поділяються на функціональна та нефункціональні.

Функціональні вимоги – це вимоги, що описують те, як система має поводитися, які завдання виконувати для того, щоб виконувати свою функцію. Нефункціональні вимоги – визначають обмеження система та не впливають на функціональність програми. Зазичай вони визначають надійність та безпеку системи, технічне обслуговування, тощо[17].

Для проєктування клієнтської сторони сформовано наступні функціональні вимоги.

Навігація:

1) Користувач може перейти до головної сторінки за допомогою відповідної кнопки.

2) Користувач може перейти до сторінки календарю за допомогою відповідної кнопки.

3) Користувач може перейти до сторінки списку справ за допомогою відповідної кнопки.

4) Користувач може перейти до сторінки додавання нового клієнта за допомогою відповідної кнопки.

Головна сторінка:

1) Користувач може переглядати дані записаних клієнтів.

2) Користувач може перейти до інших сторінок проєкту.

Календар:

1) Користувач може переглядати календар та події у ньому.

2) Користувач може додавати нові події.

3) Користувач може переносити події на інший час.

4) Користувач може видаляти події з календарю.

Список справ:

1) Користувач може додавати нові справи у список справ.
2) Користувач може переглянути додані справи.
3) Користувач може помітити справи як виконані.
4) Користувач може відмітити виконану справу як не виконану.
5) Користувач може видалити справу зі списку справ.

Сторінка додавання нового клієнта:

1) Користувач може ввести ім’я нового клієнта.
2) Користувач може ввести вік нового клієнта.
3) Користувач може ввести країну проживання нового клієнта.
4) Користувач може обрати тип нового клієнта.

3.2 Налаштування та розробка інформаційної системи

Для створення проєкту використовується команда npx:

C:\Polina>npx create-react app calendario

Для початку роботи над проєктом, необхідно перейти в його папку. Для цього використовується наступна команда:

C:\Polina>cd calendario

Для створення деяких елементів необхідно використовування стороні бібліотеки. Для того, щоб користуватися пакетами третіх сторін, їх необхідно попередньо завантажити та встановити за допомогою Node Package Manager – NPM. За допомогою таких сторонніх бібліотек полегшує процес створення додатку та дозволяє використовувати додаткові функції. У цьому проєкті використовуються наступні пакети:

C:\Polina>npm i react-router-dom

C:\Polina>npm i @syncfusion/ej2-react-calendars

C:\Polina>npm i @syncfusion/ej2-base

За допомогою цих сторонніх пакетів реалізується блок навігації та перехід між сторінками проєкту, а також елемент календаря.

Для запуску проєкту та відображення результатів коду необхідно ввести до терміналу:

C:\Polina>np start

У наведеному коді показано подальше використання завантаженого пакету для створення сторінки календаря у файлі Calendar.js(Рисунок 3.1).

Calendar.js

src > pages > Calendar.js > ...

```
import React from "react";
import { ScheduleComponent, ViewsDirective, ViewDirective,
    Day, Week, WorkWeek, Month, Agenda,
    Inject, Resize, DragAndDrop } from "@syncfusion/ej2-react-schedule";
import { DatePickerComponent } from "@syncfusion/ej2-react-calendars";

const Calendar = () => {
    return (
        <div className="page-container">
            <h1 className="header">Calendar</h1>
            <ScheduleComponent>
                <Inject services={[Day, Week, WorkWeek, Month, Agenda, Resize, DragAndDrop]}/>
            </ScheduleComponent>
        </div>
    )
}

export default Calendar;
```

Рисунок 3.1 – Сторінка календаря

Для стилізації деяких елементів компонента календаря бібліотеки Syncfusion необхідно звернутися до них через клас елементів напряму та переписати їх властивості(Рисунок 3.2).

```
/*----- Calendar -----*/

.e-schedule {
  margin: 15px !important;
  font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif !important;
}

.e-toolbar-items.e-tbar-pos {
  font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif !important;
}

.e-quick-popup-wrapper {
  font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif !important;
  border-radius: 10px !important;
}
```

Рисунок 3.2 – Стилізація елементів календаря

3.3 Налаштування блоку навігації

За допомогою блоку навігації користувач може переходити між сторінками з будь-якої іншої сторінки додатку. Блок навігації знаходиться на кожній сторінці і дозволяє переходити на головну сторінку, сторінку календаря, сторінку списку справ та на сторінку створення нового клієнта.

Стандартна бібліотека Create React не включає в себе маршрутизацію. Тому для реалізації цього у програмі необхідно використати бібліотеку React-router-dom. Ця бібліотека використовується для маршрутизації на стороні клієнта. Таким чином при запиті на перехід на іншу сторінку, програма не завантажує новий файл, а скоріш оновлює нову інформацію на негайно відображає необхідний інтерфейс. За допомогою такої маршрутизації забезпечується швидка взаємодія з користувачем та динамічний досвід[18].

Перед початком роботи необхідно створити сторінки на які користувач зможе переходити за допомогою блоку навігації. Як вже зазначено в функціональних вимогах, в системі мають бути присутні такі сторінки: головна сторінка, сторінка календаря, сторінка списку справ та сторінка створення нового користувача. Для структуризації файлів в проєкті для файлів сторінок створено окрему папку Pages(Рисунок 3.3).

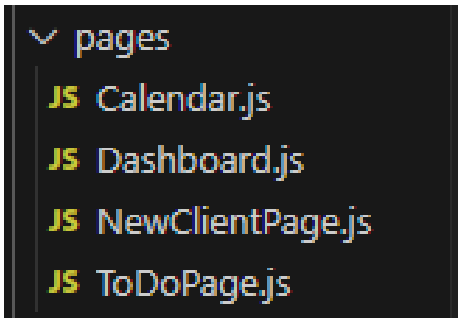

Рисунок 3.3 – Папка з сторінками програми

На початку необхідно обгорнути код, що міститься в файлі App.js в тег BrowserRouter. Цей тег забезпечує перехід між сторінками та синхронізацію

між зміною URL та зовнішнім виглядом додатку. Він використовується як батьківський тег для інших тегів[19].

Найважливішою частиною бібліотеки та основою для реалізації маршрутизації є Route. З її допомогою поєднуються сегменти URL-адрес з компонентами[20]. В Route можна передати такі властивості як path та element. В path зазначається яка саме URL-адреса призначається даному маршрутизатору, а за допомогою element вказується який саме файл чи компонент відповідає за інтерфейс. У файлі App.js маршрутизація налаштована наступним чином(Рисунок 3.4).

App.js

src > App.js > ...

```
import './App.css';
import { useState } from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Calendar from './pages/Calendar'
import ToDoPage from './pages/ToDoPage'
import Nav from './components/Nav'
import NewClientPage from './pages/NewClientPage'
import Dashboard from './pages/Dashboard';

function App() {

  return (
    <BrowserRouter>
      <Nav />
      <Routes>
        <Route path='/' element={<Dashboard />}/>
        <Route path='/dashboard' element={<Dashboard />}/>
        <Route path='/calendar' element={<Calendar />}/>
        <Route path='/todo' element={<ToDoPage/>}/>
        <Route path='/new-client' element={<NewClientPage />}/>
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

Рисунок 3.4 – Файл App.js

Після того як в файлі App.js визначено маршрутизацію, це означає, що при додаванні до URL-адреси даних, що знаходяться в властивості path, з

будь-якої частини програми можна потрапити на вказану сторінку змінюючи лише адресний рядок.

В блоку навігації така маршрутизація має відбуватися при натисканні відповідної кнопки. Для цього використовується хук useNavigate(), що також знаходиться в бібліотеці React-router-dom. Цей хук дозволяє перенаправлення на вказаний шлях URL. Використовуючи подію onClick, поточне розташування візуалізації змінюється при натисканні відповідної кнопки на сторінку, адреса якої вказана в хуці.

В програмі блок навігації реалізований наступним чином(Рисунок 3.5).

Nav.js

src > components > Nav.js > Nav

```
1   import { useNavigate } from 'react-router-dom'
2   import Logo from '../img/logo.svg'
3   import Plus from '../img/plus.svg'
4
5   const Nav = () => {
6
7       const navigate = useNavigate();
8       return (
9           <nav>
10              <div className='controls-container'>
11                  <div className='logo-container' onClick={() => navigate('/')}>
12                      <img src={Logo} alt="logo"/>
13                      <div>Calendario</div>
14                  </div>
15                  <div className='icon' onClick={() => navigate('/calendar')}>Calendar</div>
16                  <div className='icon' onClick={() => navigate('/todo')}>To-Do List</div>
17                  <div className='right-icon' onClick={() => navigate('/new-client')}>
18                      <div>Add Client</div>
19                      <img src={Plus} alt="plus"/>
20                  </div>
21
22              </div>
23          </nav>
24      )
25  }
26
27  export default Nav
```

Рисунок 3.5 – Блок навігації

Сторінки календаря та списку справ містять в кнопці лише текст, в той час як головна сторінка та сторінка додавання нового клієнту також містить картинку, що є частиною кнопки. Картинки, що використовуються, імпортовані на початку файлу та представлені у форматі svg.

3.4 Тестування системи

Для перевірки розробленої інформаційної системи управління взаємодії з клієнтами у сфері надання послуг необхідно запустити програму та перевірити взаємодію та функціональність сторінок.

При запуску програми користувач потрапляє до головної сторінки (Рисунок 3.6). Як і на всіх сторінках на ній знаходиться блок навігації та відображаються дані про клієнтів, що вже знаходяться в системі.

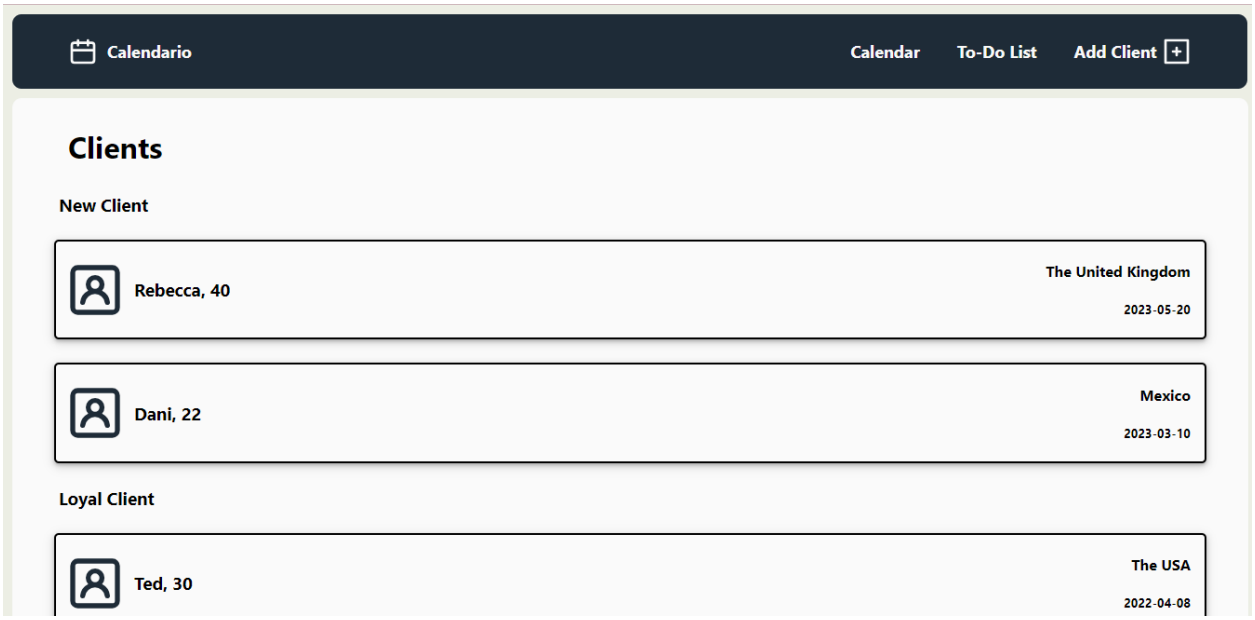

Рисунок 3.6 – Головна сторінка

На головній сторінці відображаються дані про клієнтів, що розділені за категоріями: новий клієнт (англ. New client) та постійний клієнт (англ. Loyal client). З лівої сторони відображається ім’я та вік клієнта, а з правої сторони знаходяться країна клієнта та дата першої реєстрації клієнта.

За допомогою блока навігації, при натисканні елементу Calendar, відображається сторінка взаємодії з календарем (Рисунок 3.7).

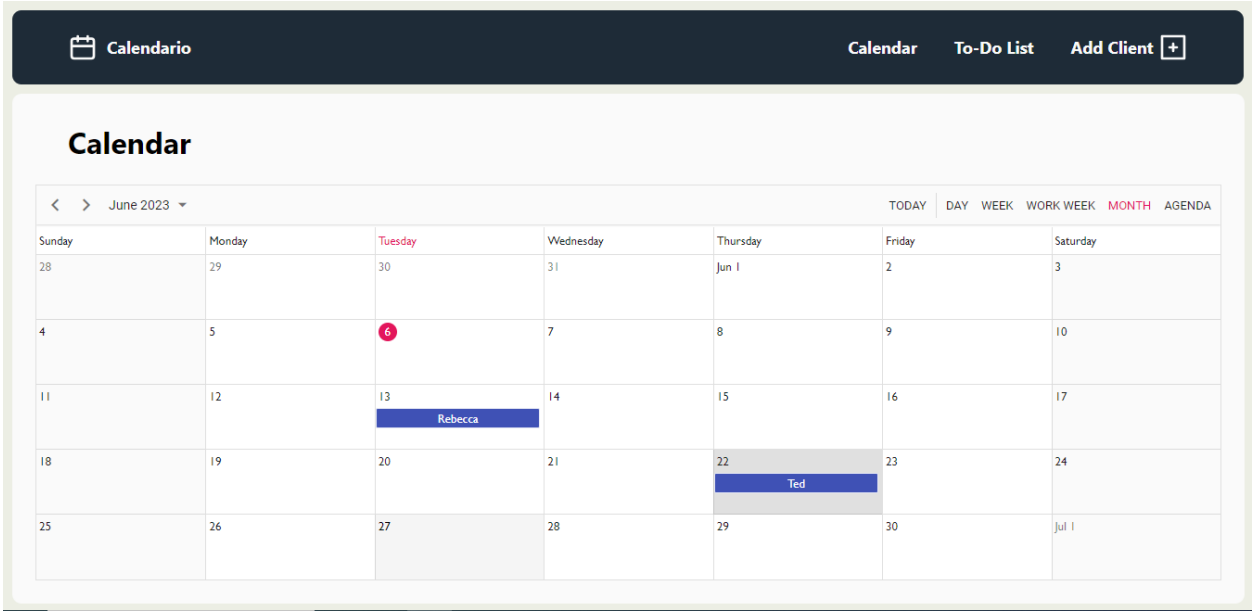

Рисунок 3.7 – Сторінка управління календарем

За допомогою календаря, що відображається на сторінці, користувач може створювати нові події, обирати час, переносити події та видаляти їх, переглядати календар у різних режимах: день, тиждень, місяць, тощо, та переглядати події, що скоро відбудяться.

При натисканні елементу To-Do List на блоці навігації, користувач перенаправляється на сторінку списку справ (Рисунок 3.8).

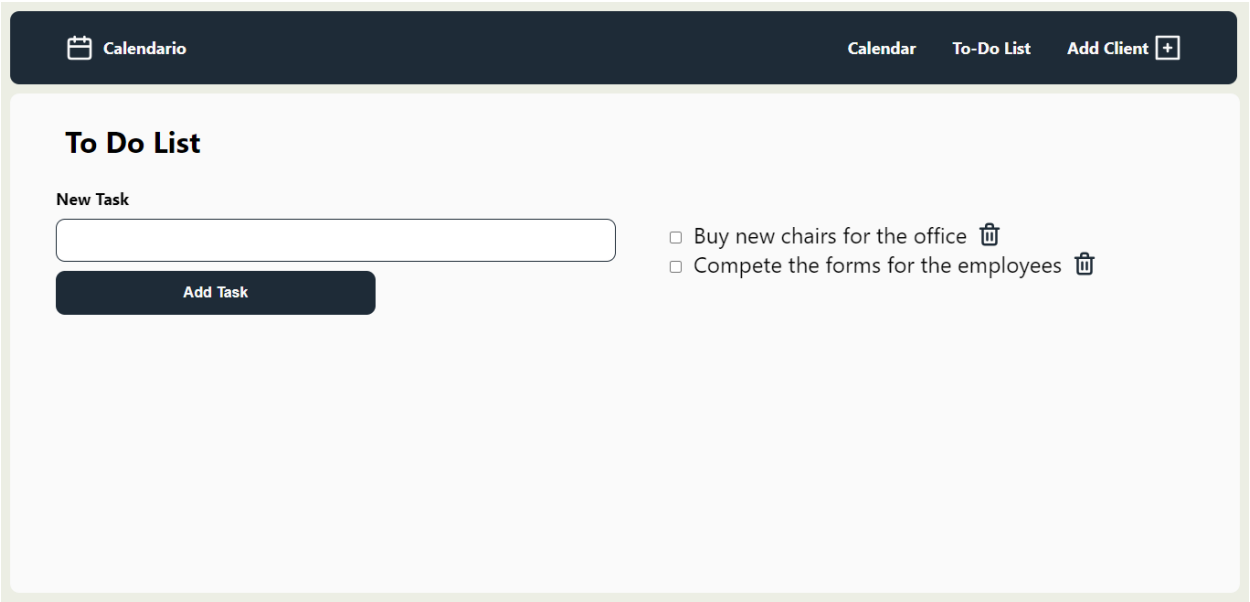

Рисунок 3.8 – Сторінка списку справ

На сторінці користувач може створити нові задачі, помітити записані задачі як виконані та видалити задачі, що більше не потрібні.

Останнім елементом, присутнім на блоці навігації, є елемент New Client. На цій сторінці користувач може ввести дані для нового клієнта за допомогою полів: ім’я клієнта, вік, країна проживання та категорію до якої належить клієнт (Рисунок 3.9).

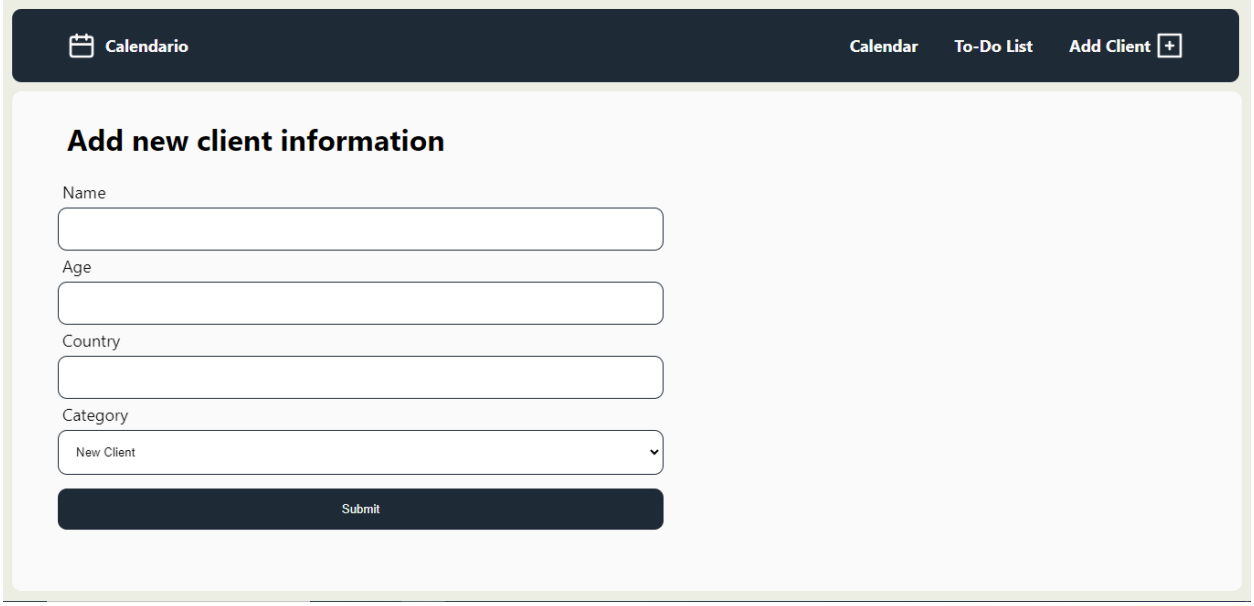

Рисунок 3.9 – Сторінка створення нового клієнта

Всі сторінки, символи, текст та інші елементи інформаційної системи відображаються коректно та відповідно до зазначених вимог. Такі елементи, як календар та список справ, повністю виконують свої функції. Навігація відбувається правильно – за допомогою визначених кнопок користувач може переходити на інші сторінки проєкту.

ВИСНОВКИ

У ході виконання переддипломної практики було виконане наступне:

1) літературний огляд джерел за тематикою інформаційні системи управління взаємодії з клієнтами у сфері надання послуг.

2) аналіз принципів створення та функціональності подібних інформаційних систем.

3) аналіз фреймворку React JS та його використання для створення веб додатків.

4) пошук та аналіз бібліотек Java Script для реалізації компонентів програми.

5) реалізація клієнтської сторони інформаційної системи управління взаємодії з клієнтами у сфері надання послуг.

6) перевірка роботи системи.

В результаті роботи було створено інформаційну систему з використанням React JS в якій користувач може створювати та змінювати події, справи та переглядати дані клієнтів, що відповідає поставленим задачам, та досліджено принципи створення клієнтського додатку та інструменти для його реалізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rose-Collins F. Статистика соціальних мереж: Соціальні медіа перемагають - і ось чому [Електронний ресурс] / Felix Rose-Collins // Ranktracker. – 2022. – Режим доступу до ресурсу: https://www.ranktracker.com/uk/blog/social-media-stats-social-media-is-taking-over-heres-why/.

2. Gerdisch E. 14 Best Jira Alternatives to Try in 2023 (Free and Paid Competitors) [Електронний ресурс] / Evan Gerdisch // ClickUp. – 2023. – Режим доступу до ресурсу: https://clickup.com/blog/jira-alternatives/.

3. Adiyanto N. Customer Relationship Management (CRM) Based On Web To Improve The Performance Of The Company. / Nurasiah Adiyanto., 2021.

4. Customer Relationship Management, CRM [Електронний ресурс] // IT Enterprise. – 2018. – Режим доступу до ресурсу: https://www.it.ua/knowledge-base/technology-innovation/customer-relationship-management-crm.

5. Пташенко О. В. Побудова CRM-системи як основи формування комунікаційної політики між організацією та кінцевим споживачем / О.В. Пташенко, Є.Д. Мірошникова // Вісник Східноукраїнського національного університету імені Володимира Даля, 2016. – № 6 (230). – С. 108-115.

6. Управління відносинами з клієнтами [Електронний ресурс] // Wikipedia. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%B2%D1%96%D0%B4%D0%BD%D0%BE%D1%81%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8_%D0%B7_%D0%BA%D0%BB%D1%96%D1%94%D0%BD%D1%82%D0%B0%D0%BC%D0%B8#%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D0%B8_CRM-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC.

7. Юрчук Н. П. CRM-системи: особливості функціонування та аналіз українського ринку / Н. П. Юрчук. – Ужгород: Науковий вісник Ужгородського національного університету, 2019. – 164 с.

8. Варіс І. О. Цифрова трансформація бізнесу: вибір, впровадження та вдосконалення CRM-систем / І. О. Варіс, О. І. Кравчук, С. А. Завгородня. – Київ, Україна, 2021.

9. JavaScript frameworks: Angular vs React vs Vue / Elar Saks., 2019. – 43 с. – Режим доступу: https://www.theseus.fi/bitstream/handle/10024/261970/Thesis-Elar-Saks.pdf

10. React [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/React.

11. Hámori F. The History of React.js on a Timeline [Електронний ресурс] / Ferenc Hámori // Rising Stack. – 2022. – Режим доступу до ресурсу: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/.

12. Patadiya J. React vs React Native: Which One to Choose and Why? [Електронний ресурс] / Jaydeep Patadiya // Radix. – 2023. – Режим доступу до ресурсу: https://radixweb.com/blog/react-vs-react-native.

13. Visual Studio Code [Електронний ресурс] // Wikipedia. – 2023. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Visual_Studio_Code.

14. Code editing. Redefined. [Електронний ресурс] // Visual Studio Code – Режим доступу до ресурсу: https://code.visualstudio.com/?wt.mc_id=DX_841432.

15. Єрємєєв Д. Створення вебзастосунку для інфографічного проектування на базі технології React / Данил Єрємєєв. – Запоріжжя: Фаховий коледж економіки та інформаційних технологій ПРАТ «ПВНЗ «ЗІЕІТ», 2022. – 92 с.

16. The World's Best UI Component Suite for Building Powerful Web, Desktop, and Mobile Apps [Електронний ресурс] // Syncfusion – Режим доступу до ресурсу: https://www.syncfusion.com/.

17. Що таке функціональні вимоги: приклади, визначення, повний посібник [Електронний ресурс] // Visure – Режим доступу до ресурсу: https://visuresolutions.com/uk/blog/functional-requirements/.

18. React Router - Getting Started [Електронний ресурс] // React Router – Режим доступу до ресурсу: https://reactrouter.com/en/main/start/overview.

19. BrowserRouter in React [Електронний ресурс] // Java Point – Режим доступу до ресурсу: https://www.javatpoint.com/browserrouter-in-react.

20. Route [Електронний ресурс] // React Router – Режим доступу до ресурсу: https://reactrouter.com/en/main/route/route.

21. Вдовиченко М. Клієнт-серверний веб-додаток для обміну зашифрованими повідомленнями / М. Вдовиченко. – Київ: Національний авіаційний університет, 2021. – 45 с. Режим доступу до ресурсу: https://dspace.nau.edu.ua/bitstream/NAU/55839/1/%d0%92%d0%b4%d0%be%d0%b2%d0%b8%d1%87%d0%b5%d0%bd%d0%ba%d0%be%20%d0%9c.%d0%a0.%20442.pdf

22. Kainu I. Optimization in React.js: Methods, Tools, and Techniques to Improve Performance of Modern Web Applications / Iida Kainu., 2022. – 19 с.

23. Іскрижицька О. Веб-додаток для адміністрування системи комунікації з використанням React JS : дипломна робота : 121 Інженерія програмного забезпечення / О. К. Іскрижицька ; керівник роботи Трунова О. В. ; Національний університет «Чернігівська політехніка», кафедра інформаційних технологій і програмної інженерії . – Чернігів, 2020. – 61 с.

24. Visual Studio Code [Електронний ресурс] // Wikipedia. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/File:Visual_Studio_Code_1.35_icon.svg.

25. What is React Router? [Електронний ресурс] // Codecademy – Режим доступу до ресурсу: https://www.codecademy.com/learn/learn-react-router/modules/learn-react-router/cheatsheet.

26. HTML DOM Events [Електронний ресурс] // W3 Schools – Режим доступу до ресурсу: https://www.w3schools.com/jsref/dom_obj_event.asp.

27. Ultimate React Router v6 Guide [Електронний ресурс] // Web Dev Simplified Blog. – 2022. – Режим доступу до ресурсу: https://blog.webdevsimplified.com/2022-07/react-router/.

ДОДАТОК А

App.js

```
import './App.css';
import { useState } from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Calendar from './pages/Calendar'
import ToDoPage from './pages/ToDoPage'
import Nav from './components/Nav'
import NewClientPage from './pages/NewClientPage'
// import LoginPage from './pages/LoginPage';
import Dashboard from './pages/Dashboard';

function App() {

  const [isLogged, setIsLogged] = useState(false);

  return (
    <BrowserRouter>
      <Nav />
      <Routes>
        <Route path='/' element={<Dashboard />}/>
        <Route path='/dashboard' element={<Dashboard />}/>
        <Route path='/calendar' element={<Calendar />}/>
        <Route path='/todo' element={<ToDoPage/>}/>
        <Route path='/new-client' element={<NewClientPage />}/>
      </Routes>
    </BrowserRouter>
  );
}
```

```
export default App;
```

App.css

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-schedule/styles/material.css";

.App {
  text-align: center;
  min-height: 100px;
  justify-content: center;
  font-family:'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans
Unicode', Geneva, Verdana, sans-serif;
}

/*----- Navbar -----*/

.controls-container {
  display: flex;
  align-items: center;
  padding: 0 50px;
```

```
  margin: 10px;
  width: 91%;
  height: 80px;
  color: #F9F9F9;
  font-weight: bold;
  font-size: large;
  background-color: #1e2b37;
  border-radius: 10px;
}

.logo-container {
  display: flex;
  align-items: center;
  margin-right: auto;
  padding: 10px;
  border-radius: 10px;
  cursor: pointer;
}

.logo-container div {
  margin-left: 10px;
}

.icon {
  padding: 10px;
  margin-left: 20px;
  border-radius: 10px;
  cursor: pointer;
}
```

```
.right-icon {
  display: flex;
  align-items: center;
  margin-left: 20px;
  padding: 10px;
  border-radius: 10px;
  cursor: pointer;
}

.right-icon div {
  margin-right: 5px;
}

.logo-container:hover, .icon:hover, .right-icon:hover {
  background-color: #22303d;
  transition: 0.7s;
}

/*----- Dashboard/Client Page -----*/

.page-container {
  margin: 10px;
  padding: 10px;
  background-color: #fafafa;
  min-height: 80vh;
  border-radius: 10px;
}
```

```
.client-container {
  margin: 10px;
}

.client-container h3 {
  margin-left: 30px;
}

.client-card {
  display: flex;
  margin: 25px;

  background-color: #fafafa;
  border-radius: 5px;
  box-shadow: 0 2px 4px 0 rgba(0, 0, 0, 0.2), 0 3px 10px 0 rgba(0, 0, 0, 0.19);
  border: 2px solid;

}

.client-left {
  display: flex;
  align-items: center;
  margin-right:auto;
  margin-left: 10px;
}

.client-left h3 {
  margin-left: 10px;
}
```

```
.client-right {
  margin-right: 15px;
  text-align: right;
}

/*----- Calendar -----*/

.e-schedule {
  margin: 15px !important;
  font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif
!important;
}

.e-toolbar-items.e-tbar-pos {
  font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif
!important;
}

.e-quick-popup-wrapper {
  font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif
!important;
  border-radius: 10px !important;
}

/*----- To Do Page -----*/

.to-do-container {
  width: 50%;
```

```
  position:fixed;
  overflow-x: hidden;
  margin-left: 20px;
}

.left {
  left: 0;
}

.right {
  right: 0;
  padding-top: 20px;
}

.header {
  margin: 10px 10px 10px 40px;
  padding: 10px;

}

.to-do-list-container {
  padding: 10px;
}

.to-do-list-container label {
  font-size: large;
  font-weight: bold;
}
```

```
.new-task {
  display: flex;
  flex-direction: column;
  margin-left: 30px;
  margin-right: 20px;
}

.new-task input {
  margin-top: 10px;
  border: 0.5px solid;
  border-color: #1e2b37;
  padding: 15px;
  border-radius: 10px;
}

.btn {
  width:350px;
  margin-top: 10px;
  margin-left: 30px;
  border: none;
  border-radius: 10px;
  background-color: #1e2b37;
  color:#F9F9F9;
  font-weight: bold;
  font-size: medium;
  padding: 15px;
}

.btn:hover {
```

```
  background-color: #22303d;
}

.list {
  list-style: none;
  margin: 15px;
  padding: 10px;
}

.list li {
  display: flex;
  align-items:center;
  margin-left: 10px;
}

.list li label{
  margin-left: 10px;
  font-size:x-large;
}

.list li:has(input:checked) .try {
  text-decoration: line-through;
}

.delete-btn {
  margin-left: 10px;
    border-radius: 5px;
    width:30px;
    height: 30px;
```

```
    color: #F9F9F9;
    border:none;
    cursor: pointer;
}

/*----- New Client -----*/

.new-client-container {
  width: 50%;
  position:fixed;
  overflow-x: hidden;
  margin-left: 20px;
}

.new-client-container form {
  display: flex;
  flex-direction: column;
  margin-left: 20px;
}

.new-client-container form label {
  padding: 5px;
  font-size: large;
}

.new-client-container form input {
  padding: 15px;
  border: 0.2px solid;
  border-color: #1e2b37;
```

```
  border-radius:10px;
}

.new-client-container form select {
  padding: 15px;
  border-color: #1e2b37;
  border-radius: 10px;
}

.submit {
  padding: 15px;
  margin-top: 15px;
  border:none;
  border-radius: 10px;
  background-color: #1e2b37;
  color: #F9F9F9;
}
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { registerLicense } from '@syncfusion/ej2-base';

// Registering Syncfusion license key
registerLicense('Mgo+DSMBaFt+QHJqVk1nQ1RbdF9AXnNOdFd0T2dabz4Nf1d
GYl9RQnZaQ1RnS3pXckxgXg==;Mgo+DSMBPh8sVXJ1S0R+X1pDaVpdX2N
LfUNxT2dbdVpzZDU7a15RRnVfRF1mSHdXdEFnXH5fdw==;ORg4AjUWIQA/
```

Gnt2VFhiQlJPcUBHQmFJfFBmQmldfVR0d0UmHVdTRHRcQltjTH9ackdnXXp
cd3M=;MjMxOTM4OEAzMjMxMmUzMDJlMzBSOWJ1TGUwem02MHo5THV
YdFBmTEhtaUQ2d21PNXRBUlNVVmVGeHJ6UDE4PQ==;MjMxOTM4OUAz
MjMxMmUzMDJlMzBIcjhZM3ovR2tTQmNQTnQrTWdGYSsrblUrdFZrUGlLQ
2VuVU9qZHFDSnFvPQ==;NRAiBiAaIQQuGjN/V0d+Xk9HfVhdWHxLflF1VW
RTe1x6cFZWACFaRnZdQV1lSXtSf0FhXXtbeXRS;MjMxOTM5MUAzMjMxM
mUzMDJlMzBHM09jOG5wQnJCMkQrbjV2b0Z6enZsdUVFRi9YWjdpNDZjSD
c0VHdJMllnPQ==;MjMxOTM5MkAzMjMxMmUzMDJlMzBqOGNWanFnSmw
ySTBZNGxndVM5Ly9YV1VFY3RpaTNUR0MrMmtsVHNHVE13PQ==;Mgo+
DSMBMAY9C3t2VFhiQlJPcUBHQmFJfFBmQmldfVR0d0UmHVdTRHRcQltjT
H9ackdnXXlaeXM=;MjMxOTM5NEAzMjMxMmUzMDJlMzBpeE1BcGhvK1pt
aERxMzdVTDRrUTB5RjZuOFozR3pSU3AvNUhvU0NMMytjPQ==;MjMxOTM
5NUAzMjMxMmUzMDJlMzBuODlaK2NxSEZJU2VsOFZGbHUwazcvMTJUT3
JrQmJQZnBObzR4VzBsc3lFPQ==;MjMxOTM5NkAzMjMxMmUzMDJlMzBH
M09jOG5wQnJCMkQrbjV2b0Z6enZsdUVFRi9YWjdpNDZjSDc0VHdJMllnPQ=
=');

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

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;
  background-color: #eceee4;
}

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

Dashboard.js

```
import { useEffect, useState } from "react"
import { Navigate } from "react-router-dom";
import ClientCard from "../components/ClientCard";

const Dashboard = () => {

    const clients =[
        {
            category: 'New Client',
            name: 'Rebecca',
            age: 40,
            country: 'The United Kingdom',
            timestamp: '2023-05-20'
        },
        {
            category: 'Loyal Client',
            name: 'Ted',
```

```
            age: 30,
            country: 'The USA',
            timestamp: '2022-04-08'
        },
        {
            category: 'New Client',
            name: 'Dani',
            age: 22,
            country: 'Mexico',
            timestamp: '2023-03-10'
        },
    ]

    const uniqueCategories = [
        ...new Set (clients?.map(({category}) => category))
    ]

    return (
        <div className="page-container">
            <h1 className="header">Clients</h1>
            <div className="client-container">
                {clients && uniqueCategories?.map((uniqueCategory,categoryIndex)
=> (
                    <div key={categoryIndex}>
                        <h3>{uniqueCategory}</h3>
                        {clients.filter(client => client.category === uniqueCategory)
                            .map((filteredClient, clientIndex) => (
                                <ClientCard
                                id={clientIndex}
```

```
                        client={filteredClient}
                        />
                    ))
                }
            </div>
        ))}
    </div>
  </div>
  )
}

export default Dashboard;
```

Calendar.js

```
import React from "react";
import { ScheduleComponent, ViewsDirective, ViewDirective, Day, Week,
WorkWeek, Month, Agenda, Inject, Resize, DragAndDrop } from
"@syncfusion/ej2-react-schedule";
import { DatePickerComponent } from "@syncfusion/ej2-react-calendars";

const Calendar = () => {
  return (
    <div className="page-container">
      <h1 className="header">Calendar</h1>
      <ScheduleComponent>
        <Inject services={[Day, Week, WorkWeek, Month, Agenda, Resize,
DragAndDrop]}/>
      </ScheduleComponent>
    </div>
```

```
    )
}

export default Calendar;
```

ToDoPage.js

```
import { useState } from "react";
import DeleteLogo from "../img/delete.svg"

const ToDoPage = () => {
    const [newTask, setNewTask] = useState("")
    const [todos, setTodos] = useState([])

    const handleSubmit = (e) => {
        e.preventDefault();

        setTodos(currentTodos => {
            return [
                ...currentTodos,
                {id: crypto.randomUUID(), title: newTask, completed: false},
            ]
        })

        setNewTask("")
    }

    const toggleTodo = (id, completed) => {
        setTodos(currentTodos => {
            return currentTodos.map(todo => {
```

```
            if (todo.id === id) {
                return {...todo, completed}
            }

            return todo
        })
    })
}

const deleteTodo =(id) => {
    setTodos(currentTodos => {
        return currentTodos.filter(todo => todo.id !== id)
    })
}

return (
    <div className="page-container">
        <h1 className="header">To Do List</h1>
        <div className="to-do-container left">
        <form onSubmit={handleSubmit} className="to-do-list-container">
            <div className="new-task">
            <label htmlFor="task">New Task</label>
            <input value={newTask} onChange={(e) =>
setNewTask(e.target.value)} type="text" id="task"/>
            </div>
            <button className="btn">Add Task</button>
        </form>
        </div>
        <section className="to-do-container right">
```

```
            <ul className="list">
               {todos.length === 0 && <label>No entries</label>}
               {todos.map(todo => {
                  return <li key={todo.id}>
                  <label>
                     <input onChange={e => toggleTodo(todo.id, e.target.checked)}
                     type="checkbox"
                     checked={todo.completed}/>
                     <label className="try">{todo.title}</label>
                  </label>
                  <img src ={DeleteLogo} onClick={() => deleteTodo(todo.id)}
className="delete-btn" alt="delete"/>
               </li>
               })}

            </ul>
            </section>
         </div>
      )
}

export default ToDoPage;
```

NewClientPage.js

```
import { useState } from "react"

const NewClientPage = () => {
   const [formData, setFormData] = useState( {
      timeStamp: new Date().toDateString()
```

```
})

const handleSubmit = (e) => {
    e.preventDefault()
}

const handleChange= (e) => {
    const value = e.target.value
    const name = e.target.name

    setFormData((prevState) => ({
        ...prevState,
        [name]:value
    })
    )
}

const categories = ['New Client', 'Loyal Client']

return (
    <div className="page-container">
        <h1 className="header">Add new client information</h1>
        <div className="new-client-container">
            <form onSubmit={handleSubmit}>
                    <label htmlFor="name">Name</label>
                    <input
                        id="name"
                        name="name"
                        type="text"
```

```
    onChange={handleChange}
    required={true}
    value={formData.name}
/>

<label htmlFor="age">Age</label>
<input
    id="age"
    name="age"
    type="text"
    onChange={handleChange}
    required={true}
    value={formData.age}
/>

<label htmlFor="country">Country</label>
<input
    id="country"
    name="country"
    type="text"
    onChange={handleChange}
    required={true}
    value={formData.country}
/>

<label>Category</label>
<select
    name="category"
    value={formData.category}
```

```
                    onChange={handleChange}
                >
                    {categories?.map((category, _index) =>(
                        <option value={category}>{category}</option>
                    ))}
                </select>

                <button className="submit">Submit</button>
            </form>
        </div>
    </div>
  )
}

export default NewClientPage;
```

Nav.js

```
import { useNavigate } from 'react-router-dom'
import Logo from '../img/logo.svg'
import Plus from '../img/plus.svg'

const Nav = () => {

  const navigate = useNavigate();
  return (
    <nav>
      <div className='controls-container'>
        <div className='logo-container' onClick={() => navigate('/')}>
          <img src={Logo} alt="logo"/>
```

```
                <div>Calendario</div>
            </div>
            <div className='icon' onClick={() =>
navigate('/calendar')}>Calendar</div>
            <div className='icon' onClick={() => navigate('/todo')}>To-Do
List</div>
            <div className='right-icon' onClick={() => navigate('/new-client')}>
                <div>Add Client</div>
                <img src={Plus} alt="plus"/>
            </div>

        </div>
    </nav>
  )
}

export default Nav
```

ClientCard.js

```
import Avatar from "../img/avatar.svg"

const ClientCard = ({client}) => {
  return (
    <div className="client-card">
        <div className="client-left">
            <img src={Avatar} alt="avatar"/>
            <h3>{client.name}, {client.age}</h3>
        </div>

        <div className="client-right">
```

```
                <h4>{client.country}</h4>
                <h5>{client.timestamp}</h5>
            </div>

        </div>
    )
}

export default ClientCard
```

logo.svg

```
<svg
    xmlns="http://www.w3.org/2000/svg"
    viewBox="0 0 24 24"
    id="calender"
    width="2em"
    height="2em"
    >
    <path fill="#F9F9F9" d="M19,4H17V3a1,1,0,0,0,0-
2,0V4H9V3A1,1,0,0,0,0,7,3V4H5A3,3,0,0,0,0,2,7V19a3,3,0,0,0,0,3,3H19a3,3,0,0,0,0,3-
3V7A3,3,0,0,0,0,19,4Zm1,15a1,1,0,0,0,1-1,1H5a1,1,0,0,0,1-1-1V12H20Zm0-
9H4V7A1,1,0,0,0,1,5,6H7V7A1,1,0,0,0,0,9,7V6h6V7a1,1,0,0,0,0,2,0V6h2a1,1,0,0,0,1,1,1
Z">
    </path>
</svg>
```

plus.svg

```
<svg
    xmlns="http://www.w3.org/2000/svg"
    viewBox="0 0 24 24" id="plus-square"
```

```
    width="2em"
    height="2em"
    >
    <path fill="#F9F9F9" d="M9,13h2v2a1,1,0,0,0,2,0V13h2a1,1,0,0,0,0,0-
2H13V9a1,1,0,0,0,0-
2,0v2H9a1,1,0,0,0,0,2ZM21,2H3A1,1,0,0,0,2,3V21a1,1,0,0,0,1,1H21a1,1,0,0,0,1-
1V3A1,1,0,0,0,21,2ZM20,20H4V4H20Z">
    </path>
</svg>
```

delete.svg

```
<svg
    xmlns="http://www.w3.org/2000/svg"
    viewBox="0 0 24 24" id="trash-alt"
    width="2em"
    height="2em"
>
<path fill="#1e2b37" d="M10,18a1,1,0,0,0,1-1V11a1,1,0,0,0-
2,0v6A1,1,0,0,0,10,18ZM20,6H16V5a3,3,0,0,0-3-
3H11A3,3,0,0,0,8,5V6H4A1,1,0,0,0,4,8H5V19a3,3,0,0,0,3,3h8a3,3,0,0,0,3-
3V8h1a1,1,0,0,0,0-2ZM10,5a1,1,0,0,1,1-
1h2a1,1,0,0,1,1,1V6H10Zm7,14a1,1,0,0,1-1,1H8a1,1,0,0,1-1-1V8H17Zm-3-
1a1,1,0,0,0,1-1V11a1,1,0,0,0-2,0v6A1,1,0,0,0,14,18Z">
</path>
</svg>
```

avatar.svg

```
<svg
    xmlns="http://www.w3.org/2000/svg"
```

```
    viewBox="0 0 24 24" id="user-square"
    width="4em"
    height="4em"
    >
    <path fill="#1e2b37" d="M14.81,12.28a3.73,3.73,0,0,0,1-2.5,3.78,3.78,0,0,0-
7.56,0,3.73,3.73,0,0,0,1,2.5A5.94,5.94,0,0,0,6,16.89a1,1,0,0,0,2,.22,4,4,0,0,1,7.94,
0A1,1,0,0,0,17,18h.11a1,1,0,0,0,.88-
1.1A5.94,5.94,0,0,0,14.81,12.28ZM12,11.56a1.78,1.78,0,1,1,1.78-
1.78A1.78,1.78,0,0,1,12,11.56ZM19,2H5A3,3,0,0,0,2,5V19a3,3,0,0,0,3,3H19a3,3,
0,0,0,3-3V5A3,3,0,0,0,19,2Zm1,17a1,1,0,0,1-1,1H5a1,1,0,0,1-1-
1V5A1,1,0,0,1,5,4H19a1,1,0,0,1,1,1Z">
    </path>
</svg>
```