

ОПИС КОДУ ДОДАТКУ

**TASKFLOW**

Підготував: Ружицький Д.

**ЗМІСТ**

[**Структура проекту. Дерево файлів. 1**](#_zdjhwpluunct)

[**Папка public 4**](#_x73prg6dmf32)

[Файл index.html 4](#_uo55nxcts16n)

[**Папка src/components 6**](#_2nwwxq7tdrr6)

[Компонент AddNewProject.jsx 6](#_nsnf954rbhzp)

[Компонент AddNewTODO.jsx 7](#_y6p4x9lf6iwi)

[Компонент Calendar.jsx 8](#_5vgl3icql2ww)

[Компонент EditTodo.jsx 8](#_ugvtynpet814)

[Компонент Login.jsx 9](#_if7a98b1zurb)

[Компонент Main.jsx 10](#_wkbn2uyayu2u)

[Компонент Modal.jsx 11](#_4ssiv9cdq33z)

[Компонент Next7Days.jsx 11](#_y10l5lhfm29j)

[Компонент NoTasks.jsx 12](#_y0g1j39qwdhr)

[Компонент Project.jsx 13](#_ffektgrhepqf)

[Компонент ProjectForm.jsx 13](#_uzm3gizih8m9)

[Компонент Projects.jsx 14](#_sxzacxaru22v)

[Компонент RenameProject.jsx 15](#_k1mb6hps2fnx)

[Компонент SideBar.jsx 16](#_mwepc1gc0sez)

[Компонент Todo.jsx 17](#_vkef6hrfx0wm)

[Компонент TodoForm.jsx 17](#_o1v1cts1fmq9)

[Компонент Todos.jsx 19](#_mo9eib6frwgq)

[Компонент User.jsx 19](#_b26jf5ecoqib)

[**Константи 21**](#_l1lxj0nywic4)

[**Контекстори 22**](#_fktmvymxd5om)

[**Firebase 23**](#_nkm5j6w3lo7y)

[**Хуки 24**](#_zetlamfvpp54)

[**Головний компонент App.jsx 26**](#_2hbtln8q1aya)

[**Точка входу (index.js) 28**](#_1aovcb8nisu0)

# **Структура проекту. Дерево файлів.**

**public**

|-- index.html

|-- logo192.png

**src**

|-- **assets**

| |-- **images**

| |-- avatar.png

| |-- intro1.svg

| |-- intro2.svg

| |-- intro3.svg

| |-- noTasks.svg

|-- **components**

| |-- AddNewProject.jsx

| |-- AddNewTODO.jsx

| |-- Calendar.jsx

| |-- EditTodo.jsx

| |-- Loader.css

| |-- Loader.jsx

| |-- Login.jsx

| |-- Main.jsx

| |-- Modal.jsx

| |-- Next7Days.jsx

| |-- NoTasks.jsx

| |-- Project.jsx

| |-- ProjectForm.jsx

| |-- Projects.jsx

| |-- RenameProject.jsx

| |-- SideBar.jsx

| |-- Todo.jsx

| |-- TodoForm.jsx

| |-- Todos.jsx

| |-- User.jsx

|-- **constants**

| |-- index.js

|-- **context**

| |-- index.js

|-- **firebase**

| |-- index.js

|-- **hooks**

| |-- index.js

|-- App.css

|-- App.js

|-- index.css

|-- index.js

**.gitignore**

**package-lock.json**

**package.json**

**README.md**

# **Папка public**

У React-проектах папка **public** використовується для зберігання статичних файлів, які не обробляються за допомогою Webpack і доступні безпосередньо з кореня веб сайту.

## **Файл index.html**

Цей HTML документ є типовим шаблоном для React-додатку, що розміщується у файлі public/index.html.

Ось опис кожного елемента в ньому:

<!doctype html>

Вказує браузеру, що документ написаний у HTML5.

<html lang="uk">

Відкриває HTML-документ та задає мову документа як українську (lang="uk").

<head>

Включає метадані документа, такі як кодові таблиці, посилання на зовнішні ресурси, стилі, опис і заголовок.

<meta charset="utf-8" />

Вказує браузеру, що сторінка використовує кодування UTF-8.

<link rel="icon" href="%PUBLIC\_URL%/logo192.png" />

Встановлює іконку (favicon) для вебсайту, яка відображається на вкладці браузера. %PUBLIC\_URL% буде замінено на кореневу URL додатку під час процесу збірки.

<meta name="viewport" content="width=device-width, initial-scale=1" />

Встановлює параметри viewport для правильної адаптації веб сайту на різних пристроях. Ширина встановлюється відповідно до ширини пристрою, а масштабування - 1.

<meta name="description" content="TaskFlow - more than a TODO List" />

Забезпечує короткий опис веб сайту, який використовується пошуковими системами та відображається в результатах пошуку.

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

Встановлює іконку для додавання на головний екран iOS пристроїв.

<link href="https://fonts.googleapis.com/css?family=Comfortaa:300,regular,500,600,700&display=swap" rel="stylesheet" />

Підключає шрифт Comfortaa з Google Fonts, який використовується у додатку. Вказані різні варіанти ваги шрифту (300, regular, 500, 600, 700).

<title>TaskFlow</title>

Встановлює заголовок вебсторінки, який відображається на вкладці браузера.

<body>

Основний вміст HTML-документу. В React-додатку зазвичай тут розміщується кореневий елемент, у якому буде відображатися весь React-компонент.

<div id="root"></div>

Цей div з id="root" є кореневим елементом для React-додатку. Всі React-компоненти будуть вбудовані всередині цього елемента за допомогою JavaScript. React додаток ініціалізується і монтується в цей елемент.

</body>

Закриває тіло документа.

</html>

Закриває HTML-документ.

# **Папка src/components**

Папка **src/components** у React-додатках використовується для зберігання всіх компонентів додатку.

Її основні призначення:

1. Організація коду: Розділяє додаток на менші, керовані частини.
2. Повторне використання: Компоненти можна використовувати в різних місцях додатку.
3. Ізоляція логіки: Кожен компонент відповідає за свій стан і рендеринг.
4. Масштабованість і співпраця: Полегшує співпрацю між розробниками та додає нові компоненти без порушення існуючої архітектури.

## **Компонент AddNewProject.jsx**

Цей код реалізує компонент **AddNewProject** в React, який дозволяє користувачам додавати нові проекти до Firestore бази даних.

*Опис коду:*

1. Імпорт модулів:

* Іконка AiOutlinePlus з бібліотеки react-icons.
* React і хук useState.
* Компоненти Modal та ProjectForm.
* Firebase модулі auth, firestore і методи для роботи з Firestore.

1. Стан компоненту:

* showModal: показує чи приховує модальне вікно.
* projectName: зберігає назву нового проекту.

1. Функція handleSubmit:

* Обробляє подію відправки форми.
* Перевіряє, чи введено назву проекту.
* Якщо користувач авторизований, перевіряє, чи існує проект з такою назвою в базі даних.
* Якщо проект не існує, додає новий проект до Firestore.

1. Рендеринг компоненту:

* Кнопка з іконкою AiOutlinePlus, яка відкриває модальне вікно.
* Модальне вікно Modal, яке містить форму ProjectForm для додавання нового проекту.

Цей компонент дозволяє користувачам додавати нові категорії проектів через модальне вікно, перевіряючи унікальність назви проекту перед додаванням.

## **Компонент AddNewTODO.jsx**

Цей код реалізує компонент **AddNewTODO** в React, який дозволяє користувачам додавати нові завдання до Firestore бази даних.

*Опис коду:*

1. Імпорт модулів:

* React та хуки useContext, useEffect, useState.
* Компоненти Modal та TodoForm.
* Бібліотека для роботи з датами dayjs.
* Константи calendarItems.
* Контекст TodoContext.
* Firebase модулі auth, firestore і методи для роботи з Firestore.
* Бібліотека для генерації випадкових кольорів randomColor.

1. Контекст і стани компоненту:

* Отримання проектів і вибраного проекту з контексту TodoContext.
* Стани:
* showModal: показує чи приховує модальне вікно.
* text: зберігає текст нового завдання.
* newDay: зберігає дату нового завдання.
* newTime: зберігає час нового завдання.
* todoProject: зберігає проект, до якого належить завдання.

1. Функція handleSubmit:

* Обробляє подію відправки форми.
* Перевіряє, чи введено текст завдання і чи проект не входить до calendarItems.
* Якщо користувач авторизований, створює нове завдання у Firestore з випадковим кольором, датою, часом і проектом.
* Після успішного додавання завдання, скидає стани і закриває модальне вікно.

1. Функція useEffect:

* Синхронізує вибраний проект з контексту TodoContext зі станом todoProject при його зміні.

1. Рендеринг компоненту:

* Кнопка, яка відкриває модальне вікно.
* Модальне вікно Modal, яке містить форму TodoForm для додавання нового завдання.

Цей компонент дозволяє користувачам додавати нові завдання через модальне вікно, зберігаючи їх у Firestore базі даних і забезпечуючи зручний інтерфейс для введення даних завдання.

## **Компонент Calendar.jsx**

Цей код реалізує компонент **Calendar** в React, який відображає календарні елементи і дозволяє вибирати проект зі списку.

*Опис коду:*

1. Імпорт модулів:

* Іконки BiCaretUpCircle і AiOutlineCalendar з бібліотеки react-icons.
* React і хук useContext.
* Константи calendarItems.
* Контекст TodoContext.

1. Використання контексту:

* Отримує функцію setSelectedProject з контексту TodoContext, яка дозволяє встановити вибраний проект.

1. Рендеринг компоненту:

* Основна обгортка div з класом calendar.
* Заголовок календаря з іконкою AiOutlineCalendar і текстом "Календар".
* Кнопка з іконкою BiCaretUpCircle.
* Список елементів календаря з calendarItems. Кожен елемент є div з класом item, який при кліку встановлює цей елемент як вибраний проект за допомогою setSelectedProject.

Компонент відображає календар зі списком елементів і дозволяє користувачу вибирати проекти, які вони хочуть переглядати або редагувати.

## **Компонент EditTodo.jsx**

Цей код реалізує компонент **EditTodo** в React, який дозволяє користувачам редагувати існуючі завдання у Firestore базі даних.

*Опис коду:*

1. Імпорт модулів:

* React і хуки useContext, useEffect, useState.
* Компонент TodoForm.
* Контекст TodoContext.
* Firebase модулі auth, firestore і методи для роботи з Firestore.
* Бібліотека для роботи з датами dayjs.

1. Стан компоненту:

* text: текст завдання.
* newDay: дата завдання.
* newTime: час завдання.
* todoProject: проект, до якого належить завдання.

1. Контекст:

* Отримує projects і selectedTodo з контексту TodoContext.

1. Використання useEffect для початкового налаштування:

* Якщо вибрано завдання (selectedTodo), встановлює початкові значення для text, newDay, newTime і todoProject.

1. Використання useEffect для оновлення Firestore:

* Якщо вибрано завдання (selectedTodo), ідентифікує користувача та завдання у Firestore.
* Використовує пакетне оновлення writeBatch для оновлення завдання у Firestore при зміні text, newDay, newTime або todoProject.

1. Функція handleSubmit:

* Обробляє подію відправки форми (на даний момент не виконує жодних дій, оскільки все оновлення відбувається в useEffect).

1. Рендеринг компоненту:

* Якщо завдання вибрано (selectedTodo), відображає форму редагування завдання за допомогою компонента TodoForm.

Цей компонент дозволяє користувачам редагувати існуючі завдання, синхронізуючи зміни з Firestore у реальному часі.

## **Компонент Login.jsx**

Цей код реалізує компонент **Login** в React, який дозволяє користувачам авторизуватись через Google і відображає вступний слайдер з інформацією про додаток TaskFlow.

*Опис коду:*

1. Імпорт модулів:

* Іконка FcGoogle з бібліотеки react-icons.
* React та хуки useNavigate.
* Слайдер Carousel з бібліотеки react-material-ui-carousel.
* Firebase модулі auth, firestore і методи для роботи з Firebase.
* Зображення для слайдеру (INTRO1, INTRO2, INTRO3).
* Бібліотека для роботи з датами dayjs.

1. Компонент Item:

* Приймає props і відображає зображення, заголовок та опис для кожного слайду.

1. Компонент Login:

* Ініціалізує навігацію за допомогою useNavigate.

1. Функція signInWithGoogle:

* Використовує GoogleAuthProvider для авторизації через Google.
* Після успішної авторизації перевіряє, чи існує користувач у Firestore.
* Якщо користувача не існує, створює новий документ для користувача, додає стандартні проекти і завдання у Firestore.
* Після авторизації перенаправляє користувача на головну сторінку.

1. Масив items:

* Містить об'єкти з інформацією про слайди (зображення, заголовок, опис).

1. Рендеринг компоненту:

* Відображає слайдер Carousel з елементами Item.
* Заголовок TaskFlow.
* Кнопка для авторизації через Google, яка викликає signInWithGoogle.

Цей компонент забезпечує зручний спосіб авторизації для користувачів через Google і відображає вступну інформацію про можливості додатку.

## **Компонент Main.jsx**

Цей код реалізує простий компонент **Main** в React, який служить обгорткою для інших компонентів або елементів.

*Опис коду:*

1. Імпорт React.
2. Компонент Main:

* Приймає children як пропс.
* Повертає div з класом main, всередині якого відображаються вкладені компоненти або елементи.

Цей компонент використовується для обгортання і структурування вмісту в додатку.

## **Компонент Modal.jsx**

Цей код реалізує компонент **Modal** в React, який показує модальне вікно з анімацією.

*Опис коду:*

1. Імпорт модулів:

* React та хук useRef.
* Анімаційні функції з react-spring.

1. Компонент Modal:

* Приймає children, showModal, та setShowModal як пропси.
* Використовує useRef для створення посилання на модальне вікно.
* Функція closeModal закриває модальне вікно при кліку поза його межами.
* Використовує useSpring для створення анімаційних ефектів при відображенні/приховуванні модального вікна.
* Повертає модальне вікно з анімаційним контейнером, якщо showModal істинне.

Цей компонент забезпечує модальне вікно з плавною анімацією та можливістю закриття при кліку поза його межами.

## **Компонент Next7Days.jsx**

Цей код реалізує компонент **Next7Days** в React, який відображає завдання на наступні сім днів.

*Опис коду:*

1. Імпорт модулів:

* dayjs для роботи з датами та додаткових плагінів customParseFormat та локалі uk.
* React та хуки useEffect і useState.
* Компонент Todo.

1. Налаштування dayjs:

* Встановлює українську локаль та додає плагін customParseFormat.

1. Компонент Next7Days:

* Приймає пропс todos (список завдань).
* Використовує стан weekTodos для зберігання завдань, відсортованих по днях тижня.

1. Функція useEffect:

* Обчислює дати для наступних семи днів, починаючи з сьогоднішнього дня.
* Відсортовує завдання по днях тижня, створюючи масив sortedTodosByDay.
* Оновлює стан weekTodos цим масивом.

1. Рендеринг компоненту:

* Відображає завдання для кожного з наступних семи днів.
* Відображає день тижня і кількість завдань.
* Для кожного завдання створює компонент Todo.

Цей компонент показує список завдань для наступних семи днів, групуючи їх по днях тижня і відображаючи інформацію про кожен день та кількість завдань.

## **Компонент NoTasks.jsx**

Цей код реалізує компонент **NoTasks** в React, який відображає повідомлення про відсутність завдань.

*Опис коду:*

1. Імпорт модулів:

* React.
* Зображення NOTASKS з відповідного шляху.

1. Компонент NoTasks:

* Приймає пропс todoTitle, який представляє заголовок категорії завдань.
* Відображає повідомлення про відсутність завдань з відповідним стилем і анімацією.
* Включає зображення NOTASKS і текстове повідомлення з динамічним включенням todoTitle.

Цей компонент використовується для інформування користувача про відсутність завдань у вибраній категорії та містить інструкцію щодо додавання нового завдання.

## **Компонент Project.jsx**

Цей код реалізує компонент **Project** в React, який відображає інформацію про проект та надає можливість редагування або видалення проекту.

*Опис коду:*

1. Імпорт модулів:

* Іконки AiFillDelete і BsFillPencilFill з бібліотеки react-icons.
* React та хуки useContext і useState.
* Компоненти RenameProject та Modal.
* Firebase модулі для роботи з Firestore.
* Контекст TodoContext.

1. Компонент Project:

* Приймає пропси project та edit.
* Отримує контекст TodoContext для роботи з вибраним проектом та проектом за замовчуванням.
* Використовує стан showModal для керування видимістю модального вікна.

1. Функція projectDelete:

* Видаляє проект та пов'язані завдання з Firestore для поточного користувача.
* Оновлює вибраний проект, якщо видаляється поточний вибраний проект.

1. Рендеринг компоненту:

* Відображає назву проекту, яку можна вибрати.
* Якщо edit дорівнює true, показує кнопки редагування та видалення.
* Якщо кількість завдань у проекті більша за нуль, показує їхню кількість.
* Використовує компонент Modal для редагування проекту через компонент RenameProject.

Цей компонент дозволяє користувачам вибирати, редагувати або видаляти проекти з Firestore бази даних.

## **Компонент ProjectForm.jsx**

Цей код реалізує компонент **ProjectForm** в React, який відображає форму для додавання або редагування проекту.

*Опис коду:*

1. Приймає пропси:

* handleSubmit: функція, яка обробляє подію відправки форми.
* heading: заголовок форми.
* value: поточне значення поля введення.
* setValue: функція для оновлення значення поля введення.
* setShowModal: функція для керування видимістю модального вікна.
* confirmButtonText: текст кнопки підтвердження.

1. Рендеринг форми:

* Відображає заголовок форми (heading).
* Поле введення для назви проекту, яке оновлює значення через setValue.
* Кнопка "Відміна", яка закриває модальне вікно через setShowModal.
* Кнопка "Підтвердити" з текстом confirmButtonText, яка відправляє форму.

Цей компонент забезпечує зручний інтерфейс для додавання або редагування проектів у додатку.

## **Компонент Projects.jsx**

Цей код реалізує компонент **Projects** в React, який відображає список проектів та дозволяє додавати нові або редагувати існуючі проекти.

*Опис коду:*

1. Імпорт модулів:

* Іконки з бібліотеки react-icons.
* React та хуки useContext і useState.
* Компоненти Project та AddNewProject.
* Контекст TodoContext.

1. Компонент Projects:

* Використовує стан showMenu для керування видимістю меню та стан edit для режиму редагування.
* Колір олівця (pencilColor) змінюється залежно від режиму редагування.

1. Контекст:

* Отримує projects з TodoContext.

1. Рендеринг компоненту:

* Заголовок з іконкою BsFillPaletteFill і текстом "Категорії".
* Кнопки для редагування (BsFillPencilFill), додавання нового проекту (AddNewProject) та розгортання/згортання меню (BiCaretUpCircle).
* Список проектів, де кожен проект відображається за допомогою компонента Project.

Цей компонент відображає категорії проектів, дозволяє додавати нові проекти, перемикати режим редагування та взаємодіяти з існуючими проектами.

## **Компонент RenameProject.jsx**

Цей код реалізує компонент **RenameProject** в React, який дозволяє користувачам перейменовувати існуючий проект у Firestore.

*Опис коду:*

1. Імпорт модулів:

* React та хуки useContext і useState.
* Компонент ProjectForm.
* Firebase модулі для роботи з Firestore.
* Контекст TodoContext.

1. Компонент RenameProject:

* Приймає пропси project та setShowModal.
* Використовує стан newProjectName для зберігання нового імені проекту.

Отримує selectedProject і setSelectedProject з TodoContext.

1. Функція renameProject:

* Перевіряє, чи існує користувач.
* Перевіряє, чи існує проект з новим ім'ям.
* Якщо проект з таким ім'ям не існує, оновлює ім'я проекту та пов'язані завдання у Firestore.
* Оновлює вибраний проект, якщо перейменований проект був вибраним.

1. Функція handleSubmit:

* Обробляє подію відправки форми.
* Викликає функцію renameProject.
* Закриває модальне вікно.

1. Рендеринг компоненту:

* Відображає форму ProjectForm з потрібними пропсами для редагування проекту.

Цей компонент дозволяє користувачам змінювати назву проекту, оновлюючи відповідні дані у Firestore, і забезпечує зручний інтерфейс для цього процесу.

## **Компонент SideBar.jsx**

Цей код реалізує компонент SideBar в React, який відповідає за відображення бічної панелі та обробку кліків для зняття виділення завдань.

*Опис коду:*

1. Імпорт модулів:

* React та хуки useContext, useEffect і useRef.
* Контекст TodoContext.

1. Компонент SideBar:

* Приймає пропс children, що дозволяє вкладати інші компоненти всередину бічної панелі.
* Використовує контекст TodoContext для доступу до функції setSelectedTodo.
* Використовує useRef для створення посилання на бічну панель.

1. Використання useEffect:

* Додає обробник подій click при монтуванні компонента.
* Видаляє обробник подій при демонтуванні компонента.

1. Функція handleClick:

* Викликається при кожному кліку на документ.
* Якщо клік був зроблений на бічну панель або всередині неї, скидає вибране завдання (setSelectedTodo(undefined)).

1. Рендеринг компоненту:

* Відображає бічну панель з класом sidebar і передає посилання sidebarRef.
* Відображає вкладені елементи (children).

Цей компонент забезпечує функціональність бічної панелі, яка знімає виділення завдань при кліку на неї або всередині неї, і дозволяє відображати додатковий вміст через пропс children.

## **Компонент Todo.jsx**

Цей код реалізує компонент **Todo** в React, який відображає завдання та дозволяє користувачам взаємодіяти з ним, наприклад, відзначати як виконане, видаляти або повторювати на наступний день.

*Опис коду:*

1. Імпорт модулів:

* Іконки з бібліотеки react-icons.
* React та хуки useContext і useState.
* Firebase модулі для роботи з Firestore.
* Бібліотека для роботи з датами dayjs.
* Контекст TodoContext.

1. Компонент Todo:

* Приймає пропс todos, який представляє завдання.
* Використовує стан hover для керування станом наведення миші.

1. Контекст:

* Отримує selectedTodo та setSelectedTodo з TodoContext.

1. Функції:

* handleDelete: Викликає deleteTodo і скидає вибране завдання, якщо воно було видалене.
* deleteTodo: Видаляє завдання з Firestore.
* checkTodo: Перемикає стан виконання завдання в Firestore.
* repeatNextDay: Повторює завдання на наступний день у Firestore.

1. Рендеринг компоненту:

* Відображає завдання з різними іконками та стилями залежно від його стану.
* Дозволяє відзначати завдання як виконане, видаляти його або повторювати на наступний день.
* Зміна стилю завдання при наведенні миші та відзначення його як виконаного.

Цей компонент забезпечує зручний інтерфейс для роботи із завданнями, дозволяючи користувачам керувати станом завдань, видаляти їх або повторювати.

## **Компонент TodoForm.jsx**

Цей код реалізує компонент **TodoForm** в React, який відображає форму для створення або редагування завдання.

*Опис коду:*

1. Імпорт модулів:

* Іконки з бібліотеки react-icons.
* Компоненти DatePicker, MobileTimePicker, та LocalizationProvider з @mui/x-date-pickers.
* AdapterDayjs для роботи з dayjs.
* React.

1. Компонент TodoForm:

* Приймає наступні пропси:
* handleSubmit: функція, яка обробляє подію відправки форми.
* heading: заголовок форми (за замовчуванням false).
* text, setText: значення і функція для оновлення тексту завдання.
* newDay, setNewDay: значення і функція для оновлення дати завдання.
* newtime, setNewTime: значення і функція для оновлення часу завдання.
* projects: список доступних проектів.
* showButtons: прапорець для відображення кнопок (за замовчуванням false).
* setShowModal: функція для керування видимістю модального вікна (за замовчуванням false).
* todoProjects, setTodoProjects: вибраний проект і функція для його оновлення.

1. Рендеринг компоненту:

* Використовує LocalizationProvider для локалізації дати і часу.
* Форма з класом todo-form містить:
* Заголовок форми (якщо заданий).
* Поле введення для тексту завдання.
* Секцію для нагадування (наразі це просто іконка і текст).
* Секцію для вибору дати (DatePicker).
* Секцію для вибору часу (MobileTimePicker).
* Секцію для вибору проекту.
* Якщо showButtons дорівнює true, відображає кнопки "Відміна" і "Підтвердити".

Цей компонент забезпечує зручний інтерфейс для введення деталей завдання, таких як текст, дата, час і проект, і дозволяє користувачам створювати або редагувати завдання.

## **Компонент Todos.jsx**

Цей код реалізує компонент **Todos** в React, який відображає список завдань залежно від вибраного проекту.

*Опис коду:*

1. Імпорт модулів:

* React та хук useContext.
* Компоненти Todo, Next7Days, та NoTasks.
* Контекст TodoContext.

1. Компонент Todos:

* Використовує контекст TodoContext для отримання списку завдань (todos) та вибраного проекту (selectedProject).

1. Рендеринг компоненту:

* Відображає заголовок вибраного проекту з анімацією.
* Якщо є завдання (todos.length > 0):
* Якщо вибраний проект дорівнює "Цього тижня", відображає компонент Next7Days з передачею всіх завдань.
* Інакше відображає кожне завдання за допомогою компонента Todo.
* Якщо завдань немає, відображає компонент NoTasks з передачею заголовку вибраного проекту.

Цей компонент забезпечує відображення списку завдань залежно від вибраного проекту, а також відображає повідомлення про відсутність завдань, якщо їх немає.

## **Компонент User.jsx**

Цей код реалізує компонент **User** в React, який відображає інформацію про користувача та забезпечує можливість виходу з облікового запису.

*Опис коду:*

1. Імпорт модулів:

* React.
* Логотип (avatar.png).
* Firebase модулі для автентифікації (auth, signOut).
* Хук useNavigate з react-router-dom.

1. Компонент User:

* Приймає пропси user (ім'я користувача) та photo (URL фото користувача).
* Використовує хук useNavigate для навігації після виходу з облікового запису.

1. Функція handleLogOut:

* Викликає signOut з Firebase для виходу користувача.
* Перенаправляє користувача на сторінку входу (/login) після успішного виходу.
* Виводить помилку в консоль, якщо вихід не вдається.

1. Рендеринг компоненту:

* Відображає фото користувача, якщо воно доступне, або стандартний логотип.
* Відображає ім'я користувача.
* Посилання "Вийти!" викликає handleLogOut для виходу з облікового запису.

Цей компонент забезпечує інтерфейс для відображення інформації про користувача та надає можливість виходу з облікового запису з перенаправленням на сторінку входу.

# **Константи**

Цей код описує вміст папки **constants**, яка містить константи, що використовуються в додатку.

*Опис вмісту файлу:*

1. calendarItems:

* Масив, що містить елементи календаря, які використовуються для відображення фільтрів завдань:
* "Сьогодні" – завдання на сьогодні.
* "Цього тижня" – завдання на цей тиждень.
* "Весь час" – всі завдання.

1. projects:

* Масив об'єктів, що представляють проекти:
* Кожен об'єкт містить:
* id: унікальний ідентифікатор проекту.
* name: назва проекту.
* numOfTodos: кількість завдань у проекті.

*Використання*:

1. calendarItems:

* Використовується для фільтрації та відображення завдань за обраними часовими рамками.

1. projects:

* Використовується для зберігання та відображення списку проектів разом з кількістю завдань у кожному проекті.

Ці константи використовуються в різних компонентах додатку для забезпечення єдиної та централізованої інформації про часові фільтри та проекти.

# **Контекстори**

Цей код реалізує контекст **TodoContext** для керування станом завдань та проектів у React-додатку.

*Опис компоненту TodoContextProvider та контексту TodoContext:*

1. Імпорт модулів:

* React, хук createContext, та хук useState.
* Кастомні хуки useTodos, useProjects, useFilterTodos, та useProjectsWithStats.

1. Створення контексту:

* TodoContext створюється за допомогою createContext.

1. Компонент TodoContextProvider:

* Використовується для обгортання інших компонентів, забезпечуючи їх доступом до стану завдань та проектів.
* Визначає defaultProject як "Сьогодні".
* Використовує хук useState для створення станів selectedProject та selectedTodo.
* Використовує кастомні хуки для отримання даних:
* useTodos: отримує всі завдання.
* useProjects: отримує всі проекти.
* useProjectsWithStats: додає статистику до проектів на основі завдань.
* useFilterTodos: фільтрує завдання на основі вибраного проекту.

1. Рендеринг компоненту:

* Використовує TodoContext.Provider для надання станів та функцій контексту всім дочірнім компонентам.
* Передає значення контексту:
* defaultProject: значення за замовчуванням для проекту.
* selectedProject, setSelectedProject: вибраний проект та функція для його оновлення.
* todos: відфільтровані завдання.
* projects: проекти з доданою статистикою.
* selectedTodo, setSelectedTodo: вибране завдання та функція для його оновлення.

*Використання:*

1. TodoContext:

* Контекст, який містить стан та функції для керування завданнями та проектами.

1. TodoContextProvider:

* Компонент, який обгортає інші компоненти, забезпечуючи їх доступом до TodoContext.

Цей контекст забезпечує централізоване керування станом завдань та проектів у додатку, що робить його зручним для використання в різних компонентах.

# **Firebase**

Цей код налаштовує та ініціалізує **Firebase** у React-додатку, використовуючи модулі для автентифікації та Firestore (базу даних).

*Опис коду:*

1. Імпорт модулів:

* initializeApp з firebase/app для ініціалізації Firebase.
* getAuth з firebase/auth для налаштування автентифікації.
* getFirestore, collection, doc, та onSnapshot з firebase/firestore для роботи з Firestore.

1. Конфігурація Firebase:

* Об'єкт firebaseConfig містить налаштування для підключення до вашого Firebase-проекту, включаючи apiKey, authDomain, projectId, storageBucket, messagingSenderId, та appId.

1. Ініціалізація Firebase:

* Викликає initializeApp з firebaseConfig, щоб ініціалізувати додаток Firebase.
* Зберігає результат в константі app.

1. Експорт модулів:

* auth: експорт об'єкта автентифікації, створеного за допомогою getAuth(app).
* firestore: експорт об'єкта Firestore, створеного за допомогою getFirestore(app).
* collection, doc, onSnapshot: повторний експорт цих методів з firebase/firestore.

*Використання:*

1. auth:

* Використовується для автентифікації користувачів у вашому додатку.

1. firestore:

* Використовується для взаємодії з базою даних Firestore.

1. collection, doc, onSnapshot:

* Методи для роботи з колекціями та документами у Firestore, а також для підписки на реальні зміни даних.

Цей код забезпечує налаштування та інтеграцію Firebase у вашому React-додатку, дозволяючи вам використовувати функції аутентифікації та Firestore для зберігання та синхронізації даних.

# **Хуки**

Цей код реалізує чотири кастомні React хуки для роботи з завданнями та проектами, використовуючи Firebase Firestore для зберігання даних.

*Опис кожного хука:*

1. useTodos

Цей хук отримує список завдань для поточного користувача з Firestore і повертає їх у стані todos.

* Стан:
  + todos: масив завдань.
  + Використання useEffect:
  + Підписується на зміни автентифікації користувача (auth.onAuthStateChanged).
  + Якщо користувач авторизований, підписується на зміни в колекції завдань (onSnapshot).
  + Оновлює стан todos при зміні даних.
  + При виході з системи або демонтажі компонента відписується від оновлень.

1. useFilterTodos

Цей хук фільтрує завдання залежно від вибраного проекту і повертає відфільтровані завдання.

* Приймає:
  + todos: масив завдань.
  + selectedProject: назва вибраного проекту.
  + Стан:
  + filteredTodos: масив відфільтрованих завдань.
  + Використання useEffect:
  + Фільтрує завдання залежно від вибраного проекту та оновлює стан filteredTodos.

1. useProjects

Цей хук отримує список проектів для поточного користувача з Firestore і повертає їх у стані projects.

* Стан:
  + projects: масив проектів.
  + Використання useEffect:
  + Підписується на зміни автентифікації користувача (auth.onAuthStateChanged).
  + Якщо користувач авторизований, підписується на зміни в колекції проектів (onSnapshot).
  + Оновлює стан projects при зміні даних.
  + При виході з системи або демонтажі компонента відписується від оновлень.

1. useProjectsWithStats

Цей хук додає статистику (кількість невиконаних завдань) до кожного проекту і повертає масив проектів зі статистикою.

* Приймає:
  + projects: масив проектів.
  + todos: масив завдань.
  + Стан:
  + projectsWithStats: масив проектів зі статистикою.
  + Використання useEffect:
  + Обчислює кількість невиконаних завдань для кожного проекту і оновлює стан projectsWithStats.

*Використання:*

* useTodos: Отримання всіх завдань поточного користувача.
* useFilterTodos: Фільтрація завдань за вибраним проектом.
* useProjects: Отримання всіх проектів поточного користувача.
* useProjectsWithStats: Додавання статистики (кількість невиконаних завдань) до кожного проекту.

# **Головний компонент App.jsx**

Цей код реалізує основний компонент **App** в React, який використовує Firebase для аутентифікації користувачів та React Router для навігації між сторінками.

*Опис компоненту:*

1. Імпорт модулів:

* React та хуки useState, useEffect.
* Компоненти з react-router-dom для маршрутизації (Router, Route, Routes, Navigate).
* Стилі з App.css.
* Компоненти SideBar, Main, User, AddNewTODO, Calendar, Projects, Todos, EditTodo, Login, та Loader.
* Firebase модуль для автентифікації (auth).

1. Компонент App:

* Використовує стан user для зберігання поточного користувача.
* Використовує стан loading для відстеження стану завантаження.

1. Використання useEffect:

* Підписується на зміну стану автентифікації користувача (auth.onAuthStateChanged).
* Оновлює стан user при зміні автентифікації.
* Встановлює loading в false, коли завантаження завершено.
* Відписується від зміни стану автентифікації при демонтажі компонента.

1. Рендеринг компоненту:

* Якщо loading дорівнює true, відображає компонент Loader.
* Інакше використовує Router для маршрутизації:
* Маршрут /login: Відображає компонент Login.
* Маршрут /:
* Якщо користувач авторизований (user), відображає інтерфейс додатку:
* SideBar, що містить компоненти User, AddNewTODO, Calendar, Projects.
* Main, що містить компоненти Todos, EditTodo.
* Якщо користувач не авторизований, перенаправляє на /login за допомогою Navigate.

*Використання*:

* SideBar: Відображає бічну панель з інформацією про користувача, формою для додавання нового завдання, календарем та списком проектів.
* Main: Відображає основний контент з списком завдань та формою для редагування завдань.
* Loader: Відображається під час завантаження стану автентифікації користувача.
* Login: Сторінка входу в систему.

Цей компонент забезпечує основну структуру додатку, управління станом автентифікації користувача та маршрутизацію між сторінками.

# **Точка входу (index.js)**

Цей код реалізує точку входу для React-додатку. Він використовує ReactDOM для рендерингу головного компоненту App в DOM-дерево. *Опис компоненту:*

1. Імпорт модулів:

- React і ReactDOM для роботи з React.

- index.css для загальних стилів додатку.

- App - головний компонент додатку.

- TodoContextProvider з контексту для управління станом завдань.

2. Створення кореневого елементу:

- Використовує ReactDOM.createRoot для створення кореневого елементу на основі DOM-елементу з id="root".

3. Рендеринг додатку:

- Використовує метод render для рендерингу компонентів в React.StrictMode.

- Обгортає компонент App в TodoContextProvider для надання контексту всім дочірнім компонентам.

*Опис рендерингу:*

- React.StrictMode: Активує додаткові перевірки та попередження в режимі розробки.

- TodoContextProvider: Надає контекст завдань всім компонентам всередині App.

- App: Головний компонент додатку, який містить всю логіку та інтерфейс користувача.

Цей код забезпечує налаштування контексту та рендеринг головного компоненту додатку, роблячи його доступним для всіх дочірніх компонентів.