Задание 2: Углубление в state-менеджмент и управление состоянием в приложении "Менеджер задач"

🌅 Цель:

Расширить функциональность ранее разработанного SPA-приложения "Менеджер задач" за счёт реализации полноценного state-менеджмента и улучшенного способа хранения задач. Добавить возможность создания и удаления задач, а также модернизировать структуру проекта под современные практики фронтенд-разработки.

🛠 Технические требования

1. Стек технологий:

- React + TypeScript
- React Router v6 (или любой другой инструмент для маршрутизации)
- State-менеджер: любой на выбор (Redux Toolkit / Zustand / MobX и т.д.)
- Сборщик проекта: любой на выбор (Vite / Webpack / Next.js и т.д.)
- UI-фреймворк: любой на выбор (Ant Design / Material UI / AdmiralDS), возможно самостоятельная разработка компонентов без использования UI-фреймворка
- Инструменты стилизации: на ваш выбор (CSS Modules / styled-components / tailwind и т.д.)

2. Особенности реализации:

✅ **Реализация state-менеджера**

- Переписать логику управления состоянием из первого задания с использованием выбранного state-менеджера.
- Вынести все данные о задачах в общий стор (store).
- Реализовать CRUD-операции над задачами через экшены/сеттеры/методы стора:
    - createTask
    - deleteTask
    - updateTask

✅ **Хранение данных**

- Вместо использования только React Context и локального состояния, реализовать долговременное хранение данных:
    - **Базовый уровень** : сохранение и загрузка задач из localStorage.
    - **Продвинутый уровень (по желанию)** : использовать публичный API (например, JSONPlaceholder) или подключить реальный backend (REST API, GraphQL и т.д.)

✅ **Маршруты**

- Главная страница с задачами: /
- Страница создания новой задачи: /task/new
- Страница редактирования задачи: /task/:id

✅ **Компоненты**

✅ TaskList

- Отображает список задач в виде адаптивной сетки карточек (Grid или Flexbox).
- Каждая задача отображается через компонент TaskItem.

✅ TaskItem

- Карточка задачи, содержащая:
  - Обязательный текстовый заголовок
  - Опциональное текстовое описание
  - Цветовые плашки (например, Chip/Tag) с информацией:
    - Категория: Bug / Feature / Documentation / Refactor / Test
    - Статус: To Do / In Progress / Done
    - Приоритет: Low / Medium / High
  - Дату создания задачи
  - Кнопку удаления задачи (иконка корзины или текстовая ссылка).
- По клику на компонент TaskItem должна открываться форма редактирования задачи. Форму можно реализовать как модальное окно, так и отдельную страницу по маршруту /task/:id.

✅ TaskForm

- Универсальная форма создания и редактирования задачи, открывается по маршруту /task/:id.
- Отображает полную информацию о задаче.
- Форма позволяет работать с:
  - Заголовком задачи
  - Описанием задачи
  - Категорией
  - Статусом
  - Приоритетом
  - Датой создания задачи
- Форма содержит кнопки: "Сохранить", "Отмена". Нажатие на кнопку приводит к сохранению внесенных изменений и возврату к списку задач.

✅ **Feature-Sliced Design**

- Привести структуру проекта к архитектуре Feature-Sliced Design

✅ **Навигация и UX**

- При переходе между страницами не должно быть перезагрузки (SPA поведение)
- После создания или редактирования задачи пользователь должен автоматически возвращаться на главную страницу

🔧 Рекомендации по предварительной настройке проекта

Чтобы сделать процесс разработки более комфортным и стандартизированным, рекомендуется выполнить следующую настройку перед началом реализации:

📄 tsconfig.json

Убедитесь, что в проекте есть корректно настроенный **tsconfig.json** файл.

📂 Path Aliases

Настройте алиасы для удобства импорта:

- @/ → src/
- @entities/ → src/entities/
- @features/ → src/features/

ESlint + Prettier

Добавьте в проект систему проверки кода и форматирования.

🌐 Autoprefixer & PostCSS

Добавьте в проект систему автоматического добавления vendor prefixes.

📦 Что нужно сдать?

1. **Ссылка на GitHub-репозиторий** с решением
2. **README.md** , содержащий:
    - Описание реализованного функционала
    - Инструкцию по запуску проекта
    - Используемые технологии и подходы
    - Краткое описание применённой архитектуры
1. **Живой пример** работы приложения (через Vercel, GitHub Pages и т.п.)
2. **Комментарии в коде:** добавить комментарии в формате JSDoc к интерфейсам компонентов или для объяснения каких-то нетривиальных решений

🏁 Рекомендации по оформлению

- Дизайн остаётся на ваше усмотрение — можно ориентироваться на Trello, Notion, Jira и т.д.
- Форма создания задачи может быть как на отдельной странице, так и в модальном окне
- По возможности реализуйте валидацию формы (минимально — обязательные поля)
- Можно добавить спиннер загрузки при обращении к внешнему API

🎯 Бонусные задания (на усмотрение студента)

- Добавить фильтрацию задач на главной странице по статусу, категории или приоритету
- Реализовать сортировку задач (по дате создания, приоритету и т.д.)