Level 1: Clever to-do list

Экосистема:

Требуется инициализировать базовое приложение (используя любой готовый Boilerplate вроде CRA или свои наработки), основанное на React / Vue / Angular и реализовать функционал, описанный далее. В том числе, на личном GitHub-аккаунте создать публичный репозиторий с названием этого задания (указано в заголовке документа: Innowise Lab Internship: Level 1: Clever to-do list) и возможность создавать Pull request любыми лицами.

Суть приложения: task-менеджер, позволяющий осуществлять планирование задач по дням. Приложение состоит из 3 частей:

1. Авторизация / регистрация; Sign in и Register - это 2 разные страницы, соответственно разные роуты в приложении.

2. Просмотр задач по дням; этот пункт состоит из двух основных блоков: календарь для просмотра задач и сам список.

3. Страница создания / редактирования задачи; пользователь может задать название, описание и дату задачи; кнопка “Save” / “Update” в зависимости от того, редактируем мы задачу или создаем.

В качестве ориентировочного дизайна можно использовать пример мобильного приложения, приложенный внизу описания (средний экран с заголовком Calendar реализовывать не надо). Разумеется, это приложение надо адаптировать под веб-приложение (в произвольной форме, но я бы оставлял такую же структуру и увеличивал **max-width** страницы до **762 пикселей**).

Как видно на примере приложения, под каждым днём в календаре есть 2 точки, отражающие, какого типа задачи есть в этот день: выполненные и невыполненные. Если в дне есть невыполненные задачи: показывается точка темная, если выполненные - рядом показывается точка светлее. Календарь построить по следующему образу: в горизонтальной строке показывать все оставшиеся дни текущего месяца (от сегодня и до последнего дня месяца).

Разумеется, у задач должен быть статус “done”. Checkbox слева от задачи выполняет эту функцию. **Перетаскивание задач реализовывать не надо!**

Все состояние приложения, сохранение и обновление данных, авторизацию надо реализовать через Firebase. Для выполнения задачи можно использовать личный Google-аккаунт.

Важно: все secret-значения (вроде данных о Firebase-приложении) не должны храниться в репозитории, их надо выносить в .env файл и добавлять в .gitignore репозитория.

Технические требования:

1. Использование React-Redux / Vuex для React / Vue не требуется; использование разрешается только в том числе, если вы уже знакомы с экосистемой и имели практику в управлении состоянием через внешние утилиты ранее. Для Angular - использовать сервисы для хранения данных.

2. Для роутинга в приложении в React и Vue можно использовать сторонние библиотеки. В Angular - использовать встроенный Angular Router. Важно: если пользователь не авторизовался, его не должно пустить на страницу создания или просмотра задач.

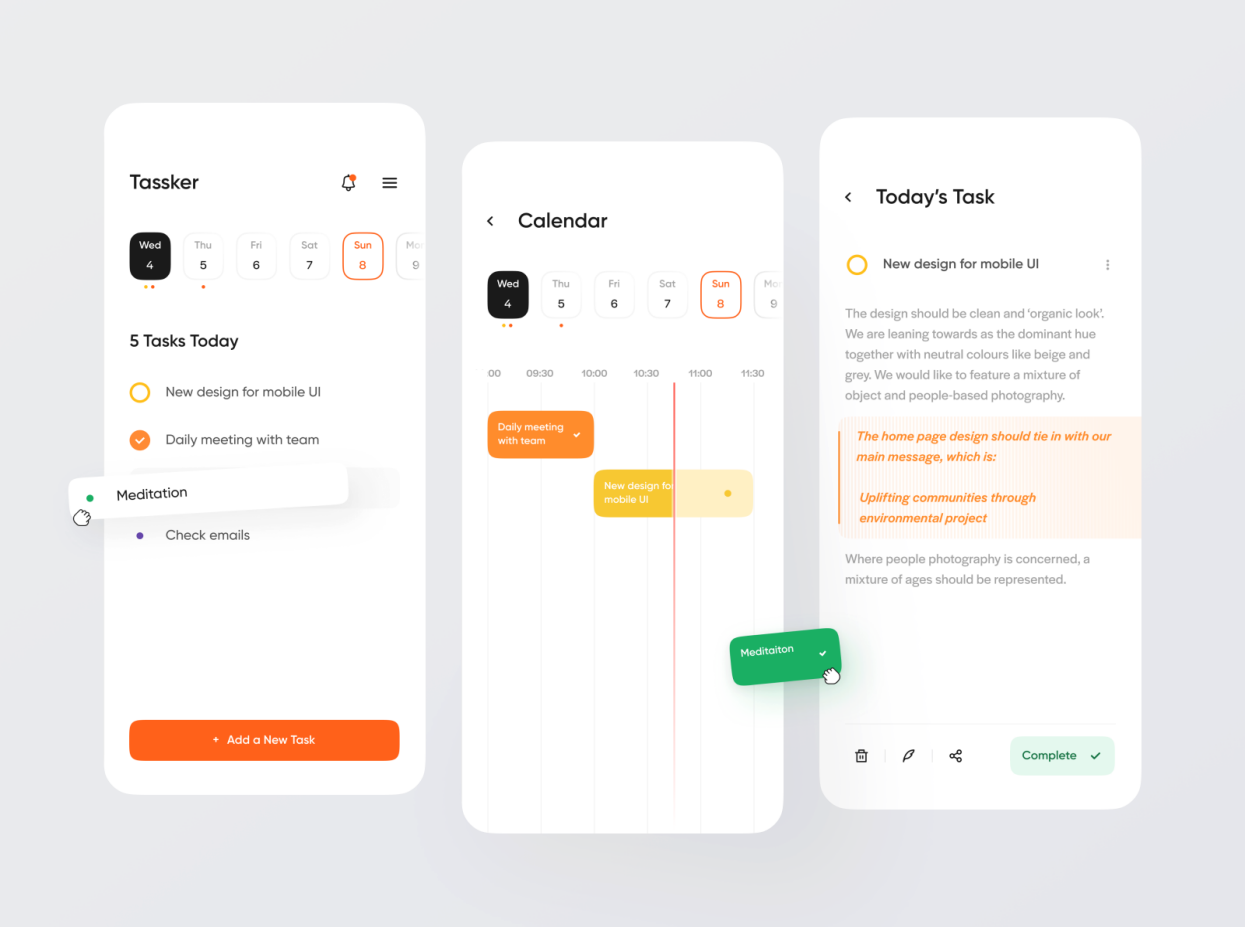
3. Перед разработкой надо настроить ESLint для приложения. В качестве дополнения можно настроить Prettier и связать его с ESLint. В том числе нужно реализовать pre-commit hook, который не позволит сделать push в репозиторий, если в приложении присутствуют ошибки ESLint.

4. Грамотная работа с базой данных. По возможности с клиента должно идти как можно меньше запросов. Банальный пример не самого оптимального использования: делать отдельный запрос на задачи для каждого дня. То есть для 30 дней месяца улетают 30 запросов. Это можно попробовать оптимизировать через Firebase Database Query.

5. Написание документации к проекту; в README-файле проекта перед сдачей задания должна быть написана краткая документация на английском или русском, состоящая из 4 пунктов: “Task” (ссылка на этот документ), “How to run the app” (инструкции по установке модулей и запуску приложения), “Database snapshot” (надо показать, как в базе данных Firebase организована работа с сущностями), и “Application stack” (описание технологического стека, использованного в приложении: какие дополнительные библиотеки и утилиты были использованы); плюсом будет написание краткого описания структуры папок: для каждой папки

описать, какого типа файлы она хранит.

6. Обработка ошибок с сервера: если пользователь пытается авторизоваться с неправильными данными - Firebase вернет ошибку, и это надо вывести пользователю на экран в виде каких-либо Toast (для этого для скорости реализации можно использовать какую-нибудь стороннюю библиотеку).



Дополнительные баллы можно получить за:

1. Действительно удобный пользовательский интерфейс.

2. Реализованный Theme-менеджмент. Возможность с легкостью поменять цветовую схему приложения **из кода** (не надо реализовывать переключатели для пользователя).

3. Реализованный “бесконечный скроллинг” в календаре. В базовой версии задания надо обеспечить пользователю возможность смотреть задачи для текущего месяца: с сегодня и до конца месяца. Плюсом будет

реализация “пагинации” со скроллингом. То есть изначально будут выводиться задачи для календарного месяца с текущего дня (например, с 22 февраля по 22 марта). При достижении скролла вправо - надо расширить диапазон на месяц вперед (до 22 апреля) и т.д.