***Checklist***

Общие требования:

* писать код максимально читаемым и понятным, соблюдая принципы, KISS, YAGNI;
* выносить повторяющиеся части кода в отдельные функции, соблюдая принцип DRY;
* соблюдать принципы SOLID, для того, чтобы программный код был более чистым и гибким;
* избегать инлайн функций (это функция, объявленная непосредственно в коде JSX компонента, она обычно используется в качестве обработчика событий или для передачи данных из родительского компонента в дочерний), т.к могут возникнуть проблемы с читаемостью, расширяемостью и оптимизацией компонентов и функций;
* давать понятные имена функциям и переменным, для того, чтобы код был более понятным и читаемым для других разработчиков и для вас в будущем;
* деструктуризировать код (извлекать значения из объекта или массива в отдельные переменные), для упрощения передачи пропсов в компоненты, уменьшения объема кода;
* избегать написания громоздких компонентов, разделять функционал на модули т.к это упростит понимание и переиспользование кода;
* использовать чистые функции внутри компонента;
* если в тестовом не используется typescript, то стоит использовать proptypes (предоставляет возможность определения типов свойств, передаваемых компоненту, что сокращает вероятность ошибок);
* не оставлять закомментированный код, в случае, если нужно что-то запомнить, можно создать документ с примерами кода;
* стрелочный вид функций является более предпочтительным, т.к такая запись обладает более удобным и компактным синтаксисом;
* избегать props drilling (передача пропсов через большое количество компонентов вниз по дереву), в таком случае можно воспользоваться Context Api или стейт менеджерами, но для небольших проектов может быть избыточно;
* использовать оптимизацию для улучшения производительности и увеличения скорости работы программы;
* избегать использования index в качестве ключа при рендеринге компонентов, стоит использовать уникальные идентификаторы для каждого элемента списка;
* хорошей практикой является использование React.Portal при работе с модальными окнами;
* для переиспользования логики, можно использовать пользовательские хуки, компоненты высшего порядка, render-props;
* для предотвращения рендеринга компонентов использовать null;
* для условного рендеринга компонентов можно использовать логический оператор &&.

Требования к стилям:

* выносить стили в отдельные файлы (в таком коде будет проще разобраться и внести правки. Для каждого компонента должна быть своя папка со стилями);
* оптимизация изображений (это поможет уменьшить их вес и увеличит скорость загрузки сайта);
* использовать styled-components или препроцессоры (это позволит расширить функционал CSS, что упростит написание кода);
* не использовать инлайн стили (такие стили сложнее поддерживать и почти невозможно переопределить);
* не использовать !important (так код будет сложнее поддерживать).